NTNU – Trondheim
Norwegian University of
Science and Technology

Customer Driven Project

Interactive Door Sign

Fall 2011

Group 7
Elise Boinnot
David Andrássy Eilertsen
Michal Krajíček
Knut Esten Melandsø Nekså
Andreas Nordahl
Sibel Ayper Taşdemir
Igoris Tirmailovas

Advisor
Muhammad Asif

Preface

TDT4290, Customer Driven Project, is claimed to be one of the most useful courses at NTNU. As students we get a unique opportunity to get to work on real-world projects with real-world customers, and because we have to work *together* as a team to succeed, this gives us even more experience and hopefully an edge when applying for jobs in the future.

The process of software system development is imperative in our project, and towards the end of it, we will hopefully have accumulated much needed knowledge in the field that we can use in the working world. Working on the documentation has taught us a great deal about how to structure, write and edit the full report of a project. This is knowledge we will definitely be able to use in future projects.

We wanted this experience to be as realistic as possible. Therefore our documentation was written without any obvious connections to the course to keep the authenticity of the project.

We would like to express our acknowledgement to our customer for his responsive approach to the changes in the project and his willingness to make compromises, as well as his feedback that helped make our system better. We would also like to thank our advisor who gave us much needed guidance and revision that shaped our documentation, and our test subjects who were a tremendous help in deciding whether or not our system met our standards.

Abstract

The amount of technology we use every day has increased since the introduction of personal computers and mobile phones. As our lives become more and more digitalized, we seek even more ways to integrate technology and ordinary everyday items. At IDI, NTNU they wish to take it a step further.

Door signs are not only a good way to inform people who the owner of an office is; they often also give information about whether or not that person is available. A potential problem with this is that even if the office owner has left a note saying when they will return, there is no way to inform people of delays. There is therefore no certain way to tell the exact time they will be coming back.

An interactive door sign could be a solution to this problem. With the added functionality of remote status updating and an interactive calendar that will inform visitors about the office owner's whereabouts and schedule, the interactive door sign could save people a lot of time that they otherwise might have spent waiting.

To create this door sign, we decided to develop a JAVA application using the newly developed SkypeKit. The application is to be run on a small USB monitor equipped with touch screen, a web camera and a microphone.

We created a fully functional prototype, implementing most of our customer's requirements. Visitors can interact with the office owner indirectly by reading their status message/schedule or by leaving them a video recorded message. They can also interact with the office owner directly by placing an audio call to the office owner's Skype account using the door sign as a means of communication. Our usability tests have shown that it is intuitive and easy to use.

The remaining requirements, and also other features, can be implemented at a later date. Due to its architecture, the application itself is flexible and can be adapted to run on a different device.

Our solution, if deemed suitable, can be used by people in the entire IDI department or even the entire university as a way to save time, and therefore also money. From there it would be easy to assume it could also be used by companies all over the world.

Trondheim, November 23, 2011

....	....
Elise Boinnot	David Andrássy Eilertsen
....	....
Michal Krajíček	Knut Esten Melandsø Nekså
....	....
Andreas Nordahl	Sibel Ayper Taşdemir
....	
Igoris Tirmailovas	

Contents

List of Figures

List of Tables

Chapter 1

Project directive

Our project directive functions as a general introduction to our project. The purpose, background and scope of the project will be presented, as will our objectives. We will then introduce the team and our project's stakeholders before we describe our team's goals and expectations for a successful project.

1.1 Project purpose and background

As written in the project description given by IDI, NTNU, "[t]he goal of this project is to design and implement a functional prototype of an interactive door sign to be used outside for instance office doors." We have been provided with a monitor that we are going to use as a door sign. This monitor should display information about the owner of the office, such as name, title and e-mail address. It should also display a customizable message that can be updated so that people will know without a doubt when the person they are looking for will be available. The name of our project is:

Interactive Door Sign

1.2 Objectives

- Create a working prototype of the application as described by the customer, with all high priority functional requirements implemented.
- The customer has mentioned two main areas to focus on: Security and usability. Different tests will ensure that the prototype meets certain standards regarding both security and usability.
- Create a broad documentation of our work and the prototype.
- Be able to present our final product.

1.3 Project scope

The project includes:

- Prototype of the application
- Documentation
- Presentation
- Small database (only for testing and presenting the application)

The project does not include:

- Hardware installation
- Hardware security

1.4 Measurement of project effects

- Utilization of the Mimo device (this monitor is too small for serious work)
- Possibility of an easy interaction between a visitor and the office owner
- Time saving for both visitors and the office owner

1.5 Duration

The project starts on August 30, 2011 and its duration will be 13 weeks. The final presentation will be on November 24, 2011.

1.6 Stakeholders

1.6.1 Group 7

The role of the project team is to fulfill all requirements given by the customer. The team must go through all phases of a typical software project, be able to provide a final report and defend the results in front of the customer and external censors. Our project realization team has 7 members. Names and contact information can be seen from table 1.1.

Name	E-mail
Michal Krajíček	michakl@stud.ntnu.no
David Andrássy Eilertsen	davidae@stud.ntnu.no
Sibel Ayper Taşdemir	tasdemir@stud.ntnu.no
Knut Esten Melandsø Nekså	knuteste@stud.ntnu.no
Andreas Nordahl	andrnord@stud.ntnu.no
Elise Boinnot	emboinno@stud.ntnu.no
Igoris Tirmailovas	igorist@stud.ntnu.no

Table 1.1: Project realization team

Organizational structure of our team is described in detail in section 2.2.

1.6.2 The customer

IDI department of NTNU university, represented by Torstein Hjelle. The customer's main concern is that the final product does what they want it to do. After all, they are the ones who are going to use it.

Name	E-mail
Torstein Hjelle	torstein.hjelle@idi.ntnu.no

Table 1.2: Customer

1.6.3 The course staff and the advisor

The course staff, meaning the course responsible and the course coordinator, is part of the evaluation team, and will take into consideration all the different factors that have played a part in our project. Our advisor, Muhammad Asif, will have followed us through our process, functioning as our *steering committee*, giving us the incentive to work harder. Collaborating with him is essential to our final report.

Name	E-mail
Muhammad Asif	muhamma@idi.ntnu.no muhammad.asif@idi.ntnu.no

Table 1.3: Advisor

1.6.4 The external examiner

It is important that the pre-delivery and the final presentation have a high quality and are as understandable as possible, showing clearly what we have accomplished. The external examiner will be part of the team that will evaluate our work.

1.7 The team

What we are mainly working towards is a good group dynamic that will allow us to create a quality product. Our final report depends on this and on how well we have understood and implemented the given requirements, as well as our overall execution of the project.

1.7.1 Goals

Our goal as a team is first of all to be able to develop a software system that complies with the requirements provided by our customer. To us, this means a functional prototype of an interactive door sign that can be used by IDI at NTNU. We also want to create accurate documentation for our work that is close to the level of those in real-world projects. This documentation will hopefully show our progress and general effort during the project.

Secondly, we want to learn more about working with team members with different backgrounds and different levels of experience as we could encounter great diversity throughout our careers.

1.7.2 Expectations of team members

This project will be a success if we are able to ...

- Learn enough from the experience to feel we can use it in future projects
- Work as a team, regardless of our different backgrounds and international students
- Understand and do what the customer wants
- Generate a good product that the customer likes
- Make a working prototype
- Get a good group dynamic and manage to work well together without too many problems

1.8 Project charter

The project charter can be found in appendix F.

Chapter 2

Planning

This chapter explains how we have organized our group, divided the project into manageable parts, and chosen our milestones. Our risks are listed here, including ways to handle them. The Lifecycle models we could choose from are covered here, as well as the one we ultimately chose.

With the inclusion of this chapter in our final report, it will be easier to show our team's planned progress during the 13 weeks we had for this project, compared to how it went in reality. The fact that we have a project plan we should follow is a test of our cooperation and organizational skills.

Every project is defined by the three dimensional target (triple constraint) and each of the dimension has its relative importance. As our project is non-commercial and has static resource allocation, the most important constraint is time. Therefore a feasible plan is crucial for the success of the whole project.

2.1 Resources

2.1.1 Human resources

The estimated workload of every team member is approximately 25 person-hours per week during 13 weeks. This makes 325 hours per person for the whole project. Our team consists of 7 members and therefore human resources in total are 2200 person-hours.

2.1.2 Technical resources

- Workstations in P15 building
- Virtual servers with software
- 500 pages of Printouts
- Copier in IT-154
- Electronic mailing list
- CVS / SVN configuration tool

2.2 Project organization

In this section we describe our different roles and responsibilities which reflect our skills and what we generally feel we can contribute with. This information was mapped early on during the first group meetings. To maintain group communication and flow, we also wrote some ground rules that all members of the team should follow. They are listed in appendix E.

2.2.1 Roles and responsibilities

- **Project manager** – Michal
  The main responsibility of the project manager is the whole project itself. The project manager should resolve conflicts, make sure that everyone is informed about the progress of the project and the work of other members. He should make sure everyone has something to do and that the team meets the deadlines. He is the leader of the group meetings and prepares agendas.
- **Customer contact** – David
  The customer contact has the responsibilities to receive and send information to the customer – he maintains communication with the customer. To avoid misunderstanding, he is the only one in contact with the customer by mail or telephone. Then, he has to relay the information to the team. He arranges and helps organize the meetings with the customer and has to write the customer summaries.
- **Advisor contact** – Sibel
  The advisor contact has the responsibility to maintain communication with the advisor. She has to send the required documents every week (weekly report).

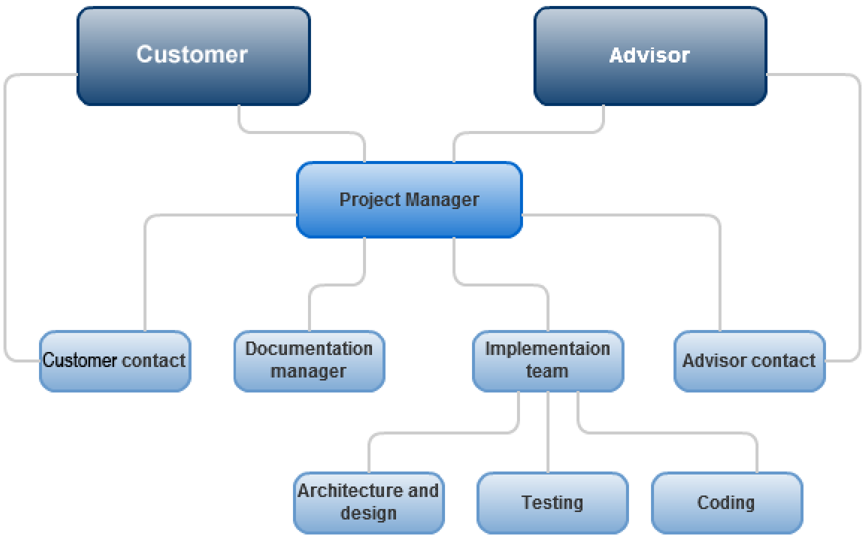


Figure 2.1: Project organization

- **Implementation manager** – Knut
  The implementation manager supervises the implementation team. He is responsible of the implementation of the software. He should divide the work between the developers.
- **Document manager** – Sibel
  The document manager is responsible for the project report. She has a good overview of what is finished and what still needs to be done for the final report. She also writes the summaries of the group meetings and advisor meetings and has to book the rooms for all meetings.
- **Architecture manager** – Igor
  The architecture manager supervises the system designer's team. He is also responsible for the architecture of the software. He should make sure that the architecture respects the model-view-controller pattern.
- **Test Manager** – Elise
  The test manager prepares the test plan – what kind of tests should be done, execute tests and monitors results.
- **Quality assurance manager** – Andreas
  The quality assurance manager should check if the implementation team respects standards in programming files (documentation for each method, indentation, etc.) and make sure that the quality of the product is high.
- **Scrum master** – David
  The Scrum master guides the team in the Scrum process.

2.2.2 Plan of meetings

We agreed on regular weekly meetings which all the group members should attend. They are held on Tuesdays from 9:15 to 12:00 .

Date	**Room**	**Week**
September 06.	R59	36
September 13.	R59	37
September 20.	R59	38
September 27.	R59	39
October 04.	R59	40
October 11.	R59	41
October 18.	K25	42
October 25.	S21	43
November 01.	S21	44
November 08.	R59	45
November 15.	R59	46
November 22.	R59	47

Moreover, we have other meetings in accordance with needs – usually 2-3 times a week.

Advisor meetings are held every Thursday from 14:15 to 15:00.

Date	**Room**	**Week**
September 15.	R54	37
September 22.	R54	38
September 29.	R54	39
October 06.	S24	40
October 13.	R54	41
October 20.	R54	42
October 27.	R54	43
November 03.	R54	44
November 10.	R54	45
November 17.	R54	46

Customer meetings are arranged in compliance with needs.

2.3 Gantt chart

We replaced the common Gantt chart with a more advanced TSTETIL diagram created in Microsoft Project 2010. This diagram shows the WBS together with a time axis (duration of particular tasks) and the logical connection among them. The TSTETIL diagram of our project can be seen in appendix C.

2.4 Milestones

Our plan contains five milestones. The first one is at the end of initiation phase containing planning, preliminary studies and requirement specification. This milestone is planned to correspond with the pre-delivery of the project report. The next two milestones are for completing the Scrum sprints. The fourth is the completion of our report and the last one is the conclusion of the project. Timing of the milestones can be seen from TSTETIL diagram in appendix C.

- Introduction & planning phase completed
- Sprint #1 completed
- Sprint #2 completed
- Documentation finished
- Project finished

2.5 Work Breakdown structure

Work breakdown structure is a method for dividing a project into smaller parts (work packages). Its purpose is to identify all packages and logically connect them. The work breakdown structure of our project can be seen on the left part of the Gantt chart.

2.6 Risk management

During our risk analysis we identified 13 potential risks to our project. The list and summary is in table 2.1. Case studies of particular risks follow.

No.	**Risk**	**Probability**	**Consequence**
1.	Inaccurate project plan	M	critical
2.	Underestimation of workload	L	critical
3.	Lack of motivation	L	significant
4.	Lack of communication internally	M	critical
5.	Lack of communication externally	M	critical
6.	Negative approach of team member(s)	L	significant
7.	Problems with the implementation	M	significant
8.	Problems with hardware	H	significant
9.	Failure of file storage (Google Documents, . . .)	L	significant
10.	Reservation issues (e.g. printer)	L	significant
11	Illness	H	minor
12.	Serious family emergency / long term illness	L	significant
13.	Interfering work / activity of a team member	H	minor

Table 2.1: Overview of risks

Inaccurate project plan

Risk number	1
Activity	Planning and monitoring
Probability	Medium
Consequences	Project team is unable to meet milestones and finish the project, stress for all members of the team
Contingency plan	Continuous monitoring of the project progress and taking appropriate measures, buffers in the plan
Responsible	Project manager

Underestimation of workload

Risk number	2
Activity	Planning and preliminary studies
Probability	Low
Consequences	Project team is unable unable to deliver a working prototype and/or report, stress for all members of the team
Contingency plan	Focus on preliminary studies, buffers in plan
Responsible	Project manager

Lack of motivation

Risk number	3
Activity	All
Probability	Low
Consequences	Project team gets behind schedule, conflicts, low quality of final product
Contingency plan	Assign responsibilities for particular phases and deliveries, maintain a good atmosphere within the team, motivate the team
Responsible	Project manager

Lack of communication internally

Risk number	4
Activity	All
Probability	Medium
Consequences	Team members will not know what their assignments and responsibilities are, unaware of the current progress, problems during the connection of software modules, lack of cooperation, overlapping work
Contingency plan	Set up team meeting on regular bases, daily scrum meetings
Responsible	Project manager

Lack of communication externally

Risk number	5
Activity	Requirements, all
Probability	Medium
Consequences	Creating the product based on bad/unapproved requirements, lack of guidance, problems during presentation of the final product
Contingency plan	Regular meetings with customer and advisor, maintain contact with customer and inform him about progress and relevant issues, signed requirements and project charter
Responsible	Customer contact

Negative approach of one or more team members

Risk number	6
Activity	All
Probability	Low
Consequences	Conflicts inside the team, lack of motivation, quality issues
Contingency plan	Motivate team members to do their work, better communication, reassign responsibilities
Responsible	Project manager

Problems with the implementation

Risk number	7
Activity	Implementation
Probability	Medium
Consequences	Delay, unable to finish prototype
Contingency plan	Obtain as much information as possible about the solution during preliminary studies and focus on the architecture, buffers in plan
Responsible	Implementation manager

Problems with hardware

Risk number	8
Activity	Preliminary studies and implementation
Probability	High
Consequences	Delay, unable to finish prototype
Contingency plan	Focus on preliminary studies, buffers in plan
Responsible	Implementation manager

Failure of file storage (Google Documents, . . .)

Risk number	9
Activity	All
Probability	Low
Consequences	Unable to deliver a working prototype and/or report, delay, stress
Contingency plan	Regular backups
Responsible	Project manager

Reservation issues (e.g. printer)

Risk number	10
Activity	Conclusion
Probability	Low
Consequences	Unable to deliver the final report, delay
Contingency plan	Include the reservation of important technical equipment to the plan, make reservation in advance
Responsible	Documentation manager

Illness

Risk number	11
Activity	All
Probability	High
Consequences	Delay, problems with communication
Contingency plan	Buffers in plans, proper documentation of work enabling it to be taken over by other team members
Responsible	Project manager

Serious family emergency/long term illness

Risk number	12
Activity	All
Probability	Low
Consequences	Delay, problems with communication
Contingency Plan	Buffers in plans, proper documentation of work enabling it to be taken over by other team members
Responsible	Project manager

Interfering work/activity of a team members

Risk number	13
Activity	All
Probability	High
Consequences	Delay
Contingency plan	Do work in advance, buffers in plans, proper documentation of work enabling it to be taken over by other team members
Responsible	Project manager

2.6.1 Risk handling

Table 2.2 shows how we approached risks. The first row describes consequences (Minor, Significant, Critical) and the first column describes probability (Low, Medium, High).

	Minor	**Significant**	**Critical**
High	monitor	take measures	take measures
Medium	neglect	monitor	take measures
Low	neglect	neglect	monitor

Table 2.2: Table for risk handling

2.7 Templates and standards

2.7.1 Templates

For the purpose of this project we created the following templates:

- Group meeting template
- Customer meeting template
- Weekly report template

Templates are stated in appendix B.

2.7.2 Standards

Dropbox

- Files should be named only in English and the name should represent the content.
- File names should be all in lower case and contain only letters and numbers. They should have underscores instead of spaces.
- Files should be structured in folders to make navigation easier.

Coding convention

Since most of the group will be involved in implementation and might have individual coding style it is write to follow more general style.

- Classes, methods, variables, etc. should be named using only English words.
- JavaDOC should be used to describe methods, classes and variables. The description should include what the method is used for with a brief explanation on how it is used, if necessary.
- Since we will all be using Eclipse, our code should be formatted before committing using source.

- Correct all errors before committing the code.

Quotation standards

We used BibTeXfor quotations. BibTeXis a tool for generating bibliography in the LaTeX environment.

2.8 Software for cooperation and planning

Because of our group members' disagreeing schedules it was essential to set up good, working and accessible remote communication lines and ways to share documents in an accessible format.

LaTeX

We chose to use LaTeXbecause it allowed us to concentrate more on the content of the document, rather than its looks. That helped us create a large, professional looking document with comfortable navigation, without using additional time to format it.

Version control

Subversion was chosen because some group members had used it before. Since all members of the implementation team are also using Eclipse IDE, it was easy to set up and maintain.

Dropbox

For file sharing, we chose to use Dropbox. Even though there are better, more powerful alternatives offering more space and functionality, such as Tonido, Wuala and Live Mesh, to mention a few, Dropbox was chosen mostly because of its popularity and the fact that most of the group members were already familiar with it. Other reasons included the lack of need in extra functions offered by the alternatives. Even though alternatives offered the same ease of use, it might have taken extra time for the group to get comfortable using them. We used Dropbox mainly to share and store more static documents and informational material.

Below is our Dropbox folder structure listing content and explanations.

- Diagrams – contains graphical use case and organizational diagrams in PNG format.
- Doc – contains all ready-to-use files for the final report and the latest version of the compiled report in PDF format.
- Meetings & Reports – contains summaries of all meetings with the advisor, customer and group. It also contains our weekly reports.
- Planning – contains a project plan in form of a Gantt chart.
- Reports from recent years – examples of other groups' final reports.
- Scrum materials – informational material on Scrum.

- Templates – templates for the customer and group meetings and the weekly report.

Google Docs

Everyone in the group worked on the final report, so to make document sharing and editing in real time by more than one team member easy and accessible, we chose to use Google Docs. It also provided us with means to share spreadsheets for filling in our weekly hours report and a way to post ideas, questions and suggestions to other group members.

Not all members of the group were familiar with LaTeX, so for the friendlier user interface we used latexlab.

Communication within the group

Our primary line of communication was e-mail. It is free, everyone uses it and it guaranties that recipients get the information as it was sent. The use of e-mail also eliminates the need to inform everyone individually. As for instant messaging, two means were chosen:

- Skype – popular and accessible to all instant messaging applications.
- IRC – popular amongst IT people and mainly used by the implementation team.

Microsoft Project 2010

We used Microsoft Project 2010 for planning purposes. We are in possession of a license for this software from the MSDN Academic Alliance program [16]. This software helped us create the WBS structure and connections among tasks as well as define milestones and the length of particular tasks and sprints. This software also helped us create a Time-scaled tasks with explicit task interdependency linkage (TSTETIL) diagram and was useful during basic tracking of our progress.

Chapter 3

Preliminary studies

In our preliminary studies we will give a short description of our customer's requirements. We will show our research on the devices we were given, and discuss various solutions on how to best implement our requirements. We will also mention components and technical constraints that must be taken into consideration. In our evaluation criteria we have written which conditions must be met in order to deem a solution optimal. This solution will then be described and justified, thus concluding the chapter.

We found it important to describe and discover new concepts, methodologies, technology and techniques necessary to succeed in the given project task. Any possible issues will be important to find as early as possible in the requirement phase. Another important point is project size; the amount of time we get to finish the project conveys the size and importance of it compared to any other course project to date.

The main focus of this chapter will be on the technical and technological constraints with the different devices. After extensive research, it was concluded that it would not be possible to implement many of the requirements on the first device given to us. Therefore we were given a new device and development for the old device was discontinued. However, research for both the devices will be presented in this chapter. This is to justify why we had to change the device, as well as show that much of the research that was done was relevant also for our new device.

3.1 The DoorSign

3.1.1 The situation and solutions of today

As of today, there is no solution that satisfies the customer's needs at NTNU. The standard door sign is a plaque, which contains the room number, name and title of the person who is occupying the office; the owner. It does not contain any kind of contact information or notifications that can easily be changed, if we disregard any post-it-like solutions. The ability to leave a message is restricted; the person must physically write them beforehand, which is impossible to do if you are not physically near the office, e.g. indisposed or late.

3.1.2 The desired solution

The wanted situation is to have an interactive door sign installed outside the office doors. This door sign will be able display customized messages via a remote device, e.g. a web-app, mobile-app, e-mail or something similar. People can also leave audio and video messages by interacting with the door sign when the owner of the office is indisposed, which the owner can then check remotely any time via an application. Streaming video and audio in real-time is also desirable as it will allow the door sign to act as an intercom, regardless of the whereabouts of the owner.

3.2 Customer requirements

All the following functional and non-functional requirements stated in this section are elaborated by the customer through the compendium, meetings and e-mail and interpreted by the project group.

3.2.1 Functional requirements

- BR1: The door sign shall display non-sensitive information about the person working in the office.
- BR2: The owner of the door sign should be able to create, edit and delete a message that shall be displayed on it, from a remote device.
- BR3: The door sign should be interactive to people wishing to contact the person in the office with a video-call button, camera, microphone and display.
- BR4: If the receiving device (to the owner of the office) is connected to the Internet, it should be able to answer calls, if nothing else is stated, by people at the door sign.
- BR5: If the receiving device (to the owner of the office) is not connected to the Internet, the caller will be able to leave a short video/audio message.
- BR6: If the owner of the office does not wish to be contacted, e.g. stated in a short message, it should not be possible to make a call to him/her. Only the option to leave a short video/audio message is available.

- BR7: When the receiving device (to the owner of the office) has established a connection to the Internet, it should receive any messages sent to him/her during the offline period with additional information, e.g. a timestamp.
- BR8: The recording of a video/audio message to the owner of the office must stop automatically after a reasonable amount of time.

3.2.2 Non-functional requirements

- Q1 (Security): You should not be able to access other parts of the system than the door sign application.
- Q2 (Usability): You should be able to learn all the system features by yourself in a short amount of time.
- Q3 (Performance): You should be able to have a normal conversation through the video conference.

3.3 Evaluation criteria

These are the criteria that must be met regarding the possible solutions [see section 3.6 possible solutions] we have researched and found as plausible solutions to our requirements, and it will justify our choice of a solution, to some extent.

1. The solution must reflect the customer's preferences
2. The solution must be approved of by the customer
3. Group 7 must unanimously agree on the solution
4. Time estimates and complexity points must be within a realistic range
5. The solution must be able to fulfill the requirements with the highest priority

3.4 Rejected device

The biggest issue in our preliminary study was definitely the analysis and complications with the initial device presented to us by the customer; the Android tablet. After three weeks of research, documented in this section and the possible solutions section, we had to conclude that it would not be possible to implement all requirments on this device. This section will justify why we made this conclusion.

3.4.1 The Android tablet

The WonderMedia WM8650 is a cheap, Chinese made tablet, running Android 2.2. It does not support all the functions that one might expect from such a device. For instance it does not support the normal Android Market, but has a Chinese rip-off instead. Another concern with the device is that it does not seem to have a Windows driver, so connecting to your Windows computer seems impossible. The touch screen on the device is also of poor quality. You have to press hard for it to register changes. Multi-touch is not supported.

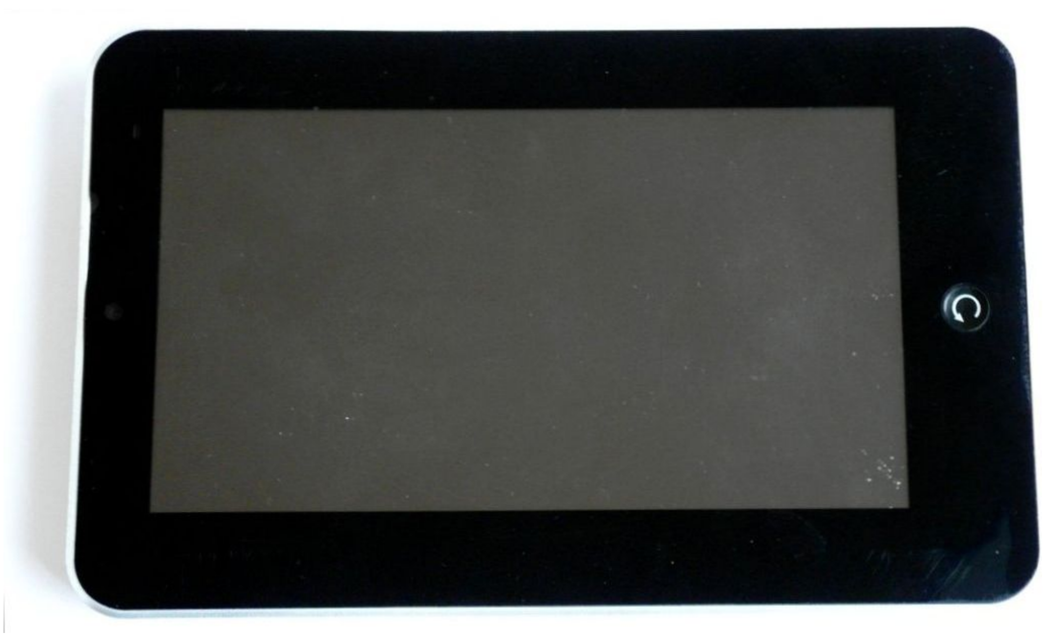


Figure 3.1: WonderMedia WM8650

Specification and details

There are no official specifications for the device. We sent an e-mail [see appendix D.2] to the company making the WM8650, but we never got a reply. The specifications below are the ones given by the seller of one of these tablets and are *not* the official specifications.

- Name: WonderMedia WM8650
- CPU: VIA 8650 800Mz
- RAM: Installed Size DDR2 256MB
- Storage: 2GB Nand Flash
- Screen: 7" touch panel (800x480)
- Audio: Built in 0.5W speaker,
- Microphone: Integrated
- Wi-Fi: Built-in 802.11B/G wireless card

Android and Android 2.2

Android is an operating system, owned by Google, developed for use mainly on tablets and mobile phones. Some of the largest mobile phone brands, like Samsung, Sony Ericsson, Motorola, LG, HTC, use Android OS for their smart phones.

Applications, such as the one we are developing, run compiled Java code on a virtual machine. This means that we mostly will be using Java code in our implementation of this application.

More specifically, we will be using a customized version of Java. This customized version of Java is very well supported by the IDE Eclipse [11], which will make our lives as developers a lot simpler. There is also a lot of sample code and tutorials published by Google, so getting familiar with general Android coding should not be a problem, even though most of our group does not have any experience with Android.

Android in general should not be much of a problem for us. However, what version of Android our tablet is able to run is of more concern. As of today it is running an old version, and there is no official update for it. The problem with this is that much of the functionality we need to complete the implementation of our requirements was first added in later versions of Android than the one our tablet is running.

The version our tablet is running is Android 2.2, at least before the update made by Hardcore-Hacker [14]. Android 2.2 does not have the SIP/VoIP library, as this was added in Android 2.3.1. Google talk and other possible solutions for the requirement concerning video conference will not be supported in this version of Android either. Because developing software for sending live video and audio over an Internet connection is out of the question, all requirements concerning video conference, concerning both audio and video, will not be possible to implement unless we are able to update the device to a later version of Android. This was later deemed impossible [see Section 3.4.1 Uberoid WM8650 HYBRiD HoneyCombMOD].

Uberoid WM8650 HYBRiD HoneyCombMOD

To improve the overall performance of the device and update it to a newer version of Android, it was decided that we would try installing a custom ROM on the tablet. After a lot of searching we have concluded that the only ROM available for this particular device is the Uberiod HYBRiD HoneyCombMOD made by HardcoreHacker [14].

Universal Uberoid WM8650 1.5.2 HoneyCombMOD v8 After installing v8 of the MOD, everything about the device got better. It was running a lot smoother and the response from it was much faster. The MOD did not update the tablet to a newer version of Android, it is still running 2.2

The drawback with installing the MOD was that the Wi-Fi no longer worked. It was able to search and detect different wireless networks but unable to connect to them properly.

Uberiod WM8650 1.5.1 HoneyCombMOD v7.1 Installing v7.1 of the MOD gave us another GUI as well as a slightly improved performance. It did not fix our issue with the wireless not working properly.

Uberoid WM8650 1.3.0 HYBRiD HoneyCombMOD v6.3 This version had worse performance than v8 and v7.1 and did not fix our Wi-Fi issue.

3.4.2 Issues and evaluation

Using the display to show a video conference should not be a problem, as long as the CPU and Wi-Fi connection are sufficiently fast. The microphone is not of high quality, but is more than good enough for us to record someone just talking. The camera has video quality that is comparable to any standard web-cam which is sufficient for our needs.

If the CPU is as good as it is suppose to be, it should not be a problem to run a video conference on it. The other requirements do not demand much of the system, as long as the device is able to connect to the Internet we should be fine. Testing the tablet with different apps and web pages gave mixed results. It was able to run games like AngryBirds [2] and Pac-Man [12], but with very low frame rate. So low in fact, that the games where not really playable. Streaming YouTube videos worked fine though. Whether the CPU performance will be a problem for us depends on if we are using the SkypeKit [10] or a open source protocol. The open source protocol choice might demand more of the system than the choosing SkypeKit for our implementation. Installing Skype [25] did not work, our device was not supported.

Wi-Fi connection seems to be the only part of the device that works well. We will be able to get a good, steady connection to the NTNU wireless network. It should not be a problem to send or receive a video and audio through the Internet connection.

We may have to disable one or more of the buttons on the device so that you can not easily access other parts of the system than intended. Disabling buttons may prove difficult without rooting it first, and rooting the device is not necessarily an easy task.

It seems the device does not have the proper drivers for us to connect it to a computer. This will prohibit effective debugging and testing of or program on the device in the implementation phase, as we may have to upload the .apk file to a web server and then download it on the device to install the program. Another possible solution is to use an SD-card to transfer the file. Rooting or installing a ROM might, at least partially, solve this problem however.

3.5 Components and technical constraints

This section will elaborate what components we are using to solve our requirements. These components will themselves limit the solution space and contain specific technical constraints we must solve or bypass, which will be described in this section.

3.5.1 The Mimo monitor

On Monday 26.09.2011 the customer presented us with a new device; a monitor. This would greatly simplify and bypass the technical constraints of the Android device. The customer, yet again, emphasized his desire for Skype to be integrated with the application.

Introduction

Mimo UM-740 is a small LCD monitor with some additional features; it has a touch screen, a microphone and a web camera. It can connected to a Windows computer through a 2.0 USB cable and be used as an external screen. This will allow us to use any available third-party software, applications, interfaces or anything similar that we may need, as long as it's supported by the Windows operating system. Therefore, the constraints will be few and limited.

Specification and details

- Name: Mimo UM-740

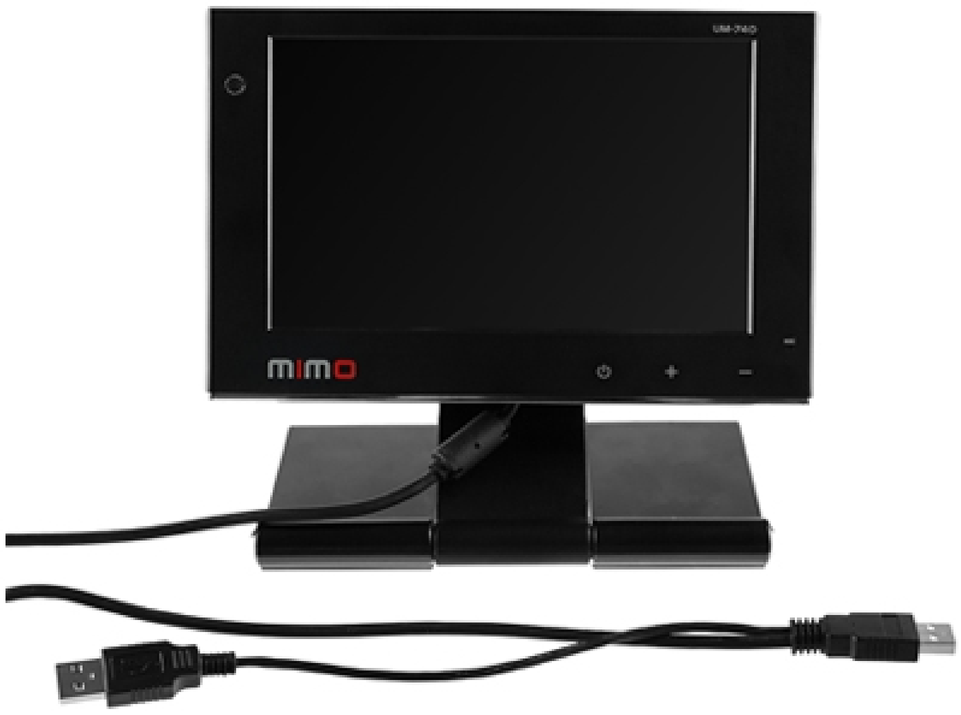


Figure 3.2: Mimo UM-740 [18]

- Screen: 7" widescreen LCD touch screen, 800 x 480 (WXGA) resolution
- Microphone: Integrated
- Camera: 1.3 megapixel Webcam
- USB: 2.0
- Other: Touch controls for on/off and volume.
- OS: Windows

Issues and evaluation

The biggest constraint is the software drivers required to utilize this device. A Windows 64-bit operating system cannot correctly install all the drivers, leaving the web camera and microphone disabled.

There are no available speakers on the Mimo monitor, which means the final software product will require an external sound source to support the audio part of the video conference.

The Mimo device is working smoothly alongside Windows 7 32-bit with all the features described in the specification.

3.5.2 SkypeKit

Introduction

SkypeKit [10] gives us access to all the functionality of Skype, including chat messages, file transfer, VoIP and video conference. We will use SkypeKit for all communication between the owner and the DoorSign. This will give the user access to the DoorSign through any device that can run Skype.

Specification and details

All Skype functionality is available through an application called SkypeKit Runtime. The DoorSign application has to communicate with the SkypeKit Runtime through a Java wrapper. Getting this to work will probably be the most challenging part. SkypeKit Runtime will take care of the actual P2P connection between the DoorSign and the owner's device.

Issues and evaluation

SkypeKit has a large amount of tutorials and a forum where you can get help if you have problems with the implementation, so we should not have too many problems using the software. As long as it gives us access to all the functionality of Skype, like it promises, we should be fine.

3.6 Possible solutions

This section presents all the solutions we have researched and tested in order to fulfill our project's requirements. These solutions, arranged in independent sub-sections, will have a short introduction, then give an overview of the services provided, which we would benefit of in the implementation phase. And finally, we will discuss any potential issues and give a short evaluation of the solution.

3.6.1 The Mimo monitor

There is only one possible solution discussed under the Mimo monitor and that is *Skype and SkypeKit*. The reason for this is that we did a lot of research on different solutions with the Android tablet [see 3.4.1]. Therefore, when presented with this device, which allowed us to use pure Java, we knew that a solution using Skype and SkypeKit would satisfy all our requirements and have limited constraints.

Skype and SkypeKit

Introduction
This was a solution that the costumer showed great interest in us implementing; an application that has Skype integrated with it. This solution would allow the owner of the office to interact with the door sign using several different devices. The only thing required by the user would be

to have a Skype account and add the door sign as a contact. Both updating the message and making a video call are things Skype would provide an intuitive and well known interface for doing. This would increase the usability of the product as no new software would be required for the user of the system, because most people working at IDI would already be familiar with, and users of, Skype.

Skype has made a developer toolkit for their program available through SkypeKit, which is a collection of software and APIs [10]. See 3.5.2 for more information about the SkypeKit.

Services Provided

- Voice chat
- Video chat/conference
- Message chat
- Calls to landline and mobile phones

Issues and evaluation
After our initial evaluation of Skype and SkypeKit as a possible solution for the Android tablet, we saw the potential in using this solution, but unfortunately, it did not work with our Android tablet [see Section 3.6.1 Skype and SkypeKit]. After receiving the Mimo monitor, we could use pure Java do develop the solution, so Skype and SkypeKit were again a possibility. The developer site contains multiple tutorials with Java and a good support team. The only issue is that this API is not open source, you have to pay 10 dollars for it, so the community is rather limited.

3.6.2 The Android tablet

We choose to include the Android tablet in this section, even though it will not be used in the final product. We found it important to show the amount of research, work and time put into this device, especially since a lot of that research was very relevant for our second device.

The following five sections will show solutions that were tested and concluded to be undesirable.

Skype and SkypeKit

Introduction
See section 3.6.1

Services Provided
See section 3.6.1

Issues and evaluation
The complication with this solution is that Skype has no available API for us to integrate with Android specifically, only with pure Java. There is a restriction on developing for mobile phones and tablets [see appendix D.1]. Since we would be using an outdated Android tablet, we concluded that this was not a good solution for us.

Session Initiation Protocol

Introduction
Session Initiation Protocol, or SIP, is a communication protocol over IP, which allows sessions like VoIP (Voice over IP), streaming and file transferring. A session could be a simple two-way telephone call or it could be a collaborative multi-media conference session. This was the best solution for our Android device after Skype failed, because it required no third-party software or similar.

Services provided
This protocol provides easy, ready made solutions for voice chat over the Internet and has a possibility of starting a video conference with other devices. SIP provides a wide variety of services that our project requires, such as voice call via Wi-Fi, video call via Wi-Fi and instant messaging for device state control.

Issues and evaluation
The main problem using SIP is that it is supported by Android SDK for Android 2.3 and up. Our device so far is only Android 2.2 compatible [see section 3.4.1 Android and Android 2.2].This restriction stops us from using SIP as our means of communication between the device and other possible client applications.

Java Sockets [19]

Introduction
Java Sockets is probably the most obvious approach to solve the requirement about updating a message through a network. It is fairly easy to implement and provides communication through a reliable TCP-connection. There are some downsides to this approach. These will be discussed later. Since this is one of the standard Java libraries, we are not dependant on any third-party interfaces or software when implementing these solutions, and there are several examples and guides available on the topic of Java Sockets.

Services provided

- Reliable TCP-connection
- Simple data transfer

Issues and evaluation
There are quite a few disadvantages of using Java Sockets. One problem is that we will have to make a client application that establishes a connection to the application on the device. Our customer was clear on the fact that he would prefer a solution where he did not have to use a custom made client-application to update the messages.

Another drawback with Java Sockets is that the server part of the Java Sockets would need a static IP-address for us to be able to connect to it. Either that, or we would have to route traffic to the port we are using through the NAT to the IP of our specific device.

There is a possibility of streaming video files (.avi,.mpeg, etc) between a server and a client by breaking down the file and/or image into datagrams at the sender's side and then rebuilding them at the receiving side. The issue here is whether or not it is possible to this with a video conference – to have a continuous bidirectional stream of data between the users through Java Sockets. Another issue is that the datagram Sockets in an Android application may be reliant on the SIP library, which is not available for Android 2.2.

Regardless, this is the only possible solution that we can with certainty implement on the Android device, but the customer, as explained in the introduction [see Section 3.6.1 Mimo Monitor], delivered a new device in hope of utilizing Skype or another third-party chat software.

Twitter

Introduction
As the selection of a third-party VoIP, streaming and a messaging service was limited and we did not find a suitable solution, we explored the possibility of separating the VoIP/streaming and messaging part.

The best solution we could find was Twitter, which was also suggested by the customer as an option to look into. Twitter is an online social networking and microblogging service that enables its users to send and read text-based posts of up to 140 characters, informally known as *tweets*.

Services provided
Twitter is a very popular and accessible service. It allows users to post tweets from anywhere with an Internet connection. Because of its popularity there are many ways to embed reading of the tweets in Android applications. The best way we found was the TWITTER4J Java lib.

- 100% pure Java. Works on any Java Platform version 1.4.2 or later
- Android compatible
- Zero dependency: No additional jars required
- Multi platform approach makes message updating a lot more comfortable and accessible

Issues and evaluation
This approach will satisfy all our message transferring requirements, but alone, it will not solve all our requirements [see Section 3.2 Customer requirements].

Google Talk [13]

Introduction
Google Talk is a free voice over Internet protocol (VoIP) and streaming client application created by Google. It uses an open protocols Jabber and/or Extensible Messaging and Presence Protocol (XMPP). It is possible for us to use our own client to connect and use the service.

Services provided

- IP voice
- File sharing
- Instant messaging
- Video conference
- Voicemail – can be 10 minutes long

To provide such services Google uses Jingle which is an extension of XMPP. XMPP is an open-standard communications protocol. To incorporate it with Google Talk, a library has been developed; libjingle (BSD license).

Issues and evaluation
Google Talk support is integrated in Android devices, but voice and video calls work only on Android 2.3.4. As our device can only run Android 2.2 [see Section 3.4.1 Android and Android 2.2] this is not a suitable solution for us.

"Google Talk with video and voice chat will gradually roll out to Nexus S devices in the next few weeks as part of the Android 2.3.4 over-the-air update and will launch on other Android 2.3+ devices in the future." [8]

Adobe LiveCycle Collaboration Service [1]

Introduction
Adobe LiveCycle Collaboration Service, formerly Adobe Flash Collaboration Service, is a hosted service that enables organizations to incorporate real-time multi-user collaboration into rich Internet applications (RIAs) that drive revenue and improve customer acquisition and retention. It provides a ready made solution with a comfortable SDK. Adobe provides communication using its new and improved RTMFP. Real Time Message Flow Protocol (RTMFP) is a UDP-based protocol that enables peer-to-peer communication between Flash Player or AIR clients. The most important features of RTMFP include low latency, end-to-end peering capability, security and scalability.

Services provided

- VoIP
  Improve productivity by enabling multiple users in dispersed teams to speak to one another through a browser without the need to install additional software.
- Web cam video broadcasts
  Enable meeting hosts and participants to use a web cam to broadcast live video in meetings, enhancing the effectiveness of overall communication. Hosts and participants can pause, share, or stop their cameras at any time during the meeting.
- Text chat
  Leverage the flexibility to conduct one-on-one or community chat in-context within an existing or new application interface without having to launch or run separate desktop chat applications.

Issues and evaluation
The service is not free. Usage of the compiled software is free of charge and does not require any additional software that is not free. Development of the software is also possible using open source software. The only fee Adobe takes is for server traffic. All servers get 15$ worth of traffic free each month, but we are not sure that traffic is enough for multiple devices. There are no guides or tutorials for Android development, and we do not even know if it is possible.

3.7 Chosen solution

The most important factor in deciding which solution to choose is whether or not it fulfills our evaluation criteria [see Section 3.3 Evaluation criteria], or comes close to fulfilling each point.

Even while we were working on the Android device we were given at the start of the project, our customer expressed a wish to integrate Skype, or a similar third-party chat and video service, in the final solution. It would be easier and more efficient to develop a system which seamlessly integrated Skype's video and audio features than to create the same features from scratch. Since the full project would have to be finished in 13 weeks, creating the whole system from scratch would therefore not be an option.

When we determined that Skype was not a possibility and we had to cut down on some of our requirements, it led to the introduction of a new device; the Mimo monitor [see Section 3.6.1 Mimo monitor]. With this we could make a Windows based system that supported Skype and in dialogue with the customer, we ultimately agreed to use SkypeKit [see Section 3.6.1 Skype and SkypeKit] to create our prototype.

This solution fulfills every point we have in our evaluation criteria. We have our customer's approval, especially due to the fact that this is the solution they would prefer to use in the first place, and the entire team has agreed to go with SkypeKit. With our team's combined Java skills, this will give us enough time to finish our prototype, which will include our most important requirements.

Our chosen solution also adds the possibility for further development in the future.

3.8 Software process model

3.8.1 Scrum

Scrum is an agile project management framework often used in software development. There are several agile software methodologies and they all share common characteristics which surrounds an iterative and incremental development method. The agile software development process can be summarized in the Agile Manifesto [5]:

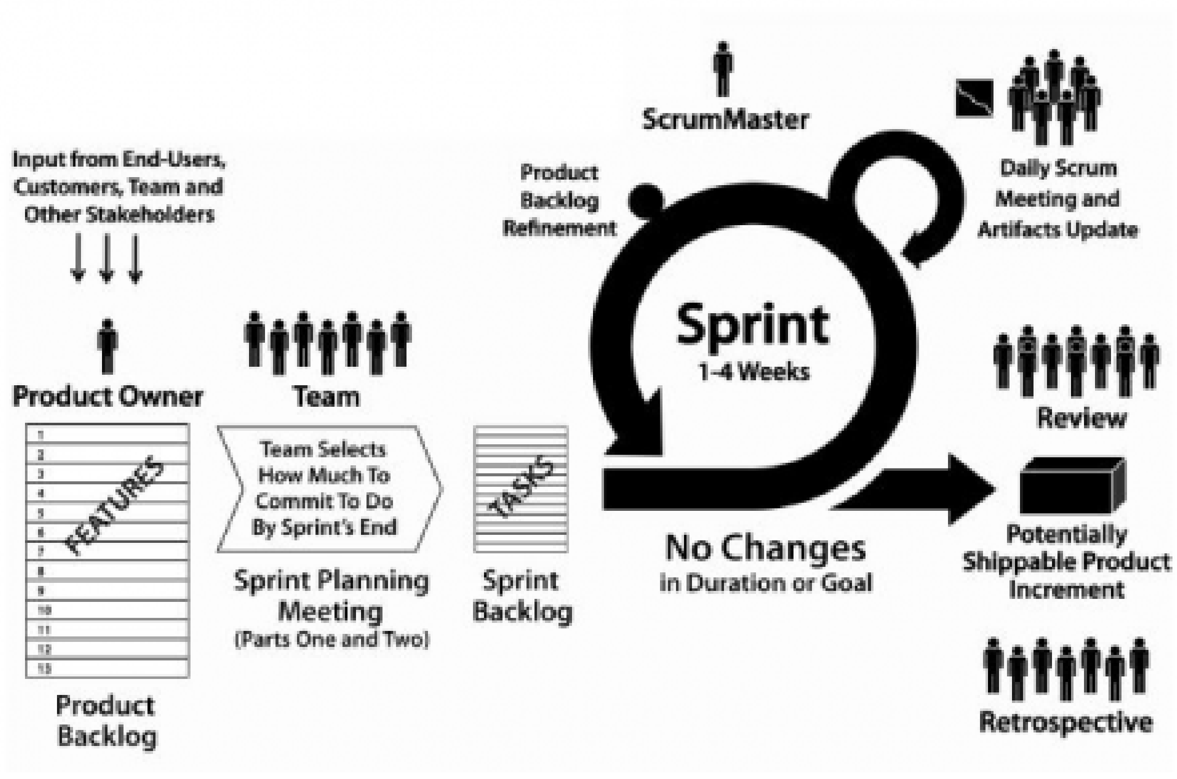
Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

By following the rules and manifesto of agile software development we see that the focus is on rapid development with incremental improvements, implementation of functionality and the constant change of requirements, risks and all other dynamic parts of the project.

Scrum is a derivation of the agile development process which comes with a set of rules, guidelines and predefined roles [see figure 3.3].

Roles

- Scrum Team – The Scrum Team is the main core of the project group, and is responsible for developing and delivering the product according to the product owner's requirements.



ScrumMaster
Input from End-Users, Customers, Team and Other Stakeholders
Product Backlog Refinement
Daily Scrum Meeting and Artifacts Update
Sprint
1-4 Weeks
Product Owner
Team
Review
FEATURES
Team Selects How Much To Commit To Do By Sprint's End
TASKS
Sprint Planning Meeting
(Parts One and Two)
Sprint Backlog
No Changes
in Duration or Goal
Potentially Shippable Product Increment
Product Backlog
Retrospective

Figure 3.3: Scrum workflow [7]

- Scrum Master – The Scrum Master is responsible for keeping the Scrum Team on track, making sure the team accomplishes the goals of each sprint, enforcing the Scrum framework and holding the daily Scrum meetings.
- Product Owner – The Product Owner conveys their vision of what the finished product should look like and how it should behave, i.e. the requirements. It is then up to the Scrum Team to produce a set of technical requirements that fulfills the Product Owner's visions.

The Process

- Product backlog – The product backlog contains a set of features (functional and non-functional requirements) elaborated by the Product Owner and specified by the Scrum Team. Here the Scrum Team prioritizes each feature (e.g. by planning poker) and receives feedback from the Product Owner.
- Sprint planning meeting – During the planning phase of the project, the features in the product backlog are distributed over a number of sprints determined by the features' priorities, complexities and the dependencies between them.
- Sprint backlog – The features of each sprint are placed in the sprint backlog of that sprint.
- Sprint – There are usually 2-4 sprints in a small to medium sized project, each allotted with a fixed amount of time. In each sprint, features are incrementally implemented into the final product. The typical steps are:
    1. Design
    2. Implementing/Coding

3. Testing
4. Integration

- Sprint review – At the end of each sprint the Product Owner, Scrum Team and Scrum Master meet up to review the completed sprint. The product backlog will often be redefined in this part of the project, as the priorities (and complexity) might shift after the review.

  A simple sprint review needs to [26]

  1. Confirm that the team has delivered on its commitments
  2. Confirm that the overall project is on track
  3. Examine the implemented functionality

- Sprint retrospective – The sprint retrospective is a team oriented self-reflection meeting focused on continuous process improvements.

3.8.2 Waterfall

The Waterfall model is a sequential software development process. Its name, Waterfall, can be interpreted as an analogy of the fundamental process steps, the phases; requirements, design, implementation, verification and maintenance [see figure 3.4].

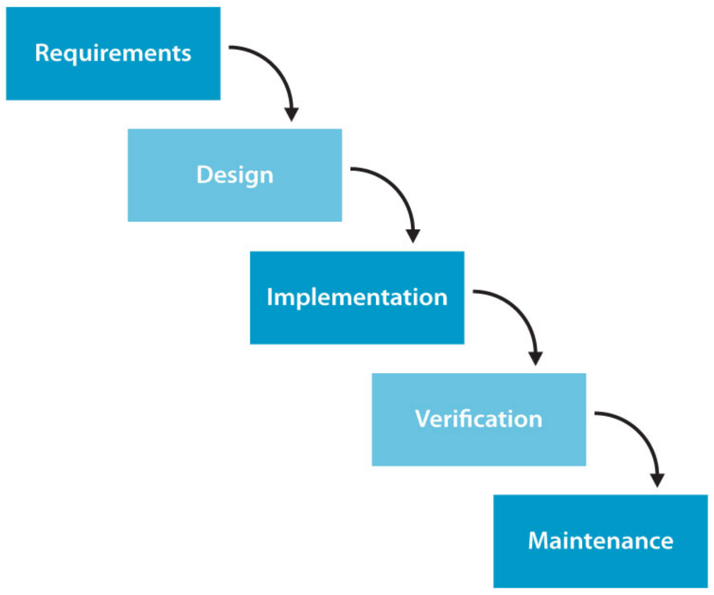


Figure 3.4: Waterfall workflow [28]

These phases are somewhat similar to the Scrum sprint phases, but there are no iterations in the Waterfall model. Each phase is only visited once through the entire project lifetime, so the flow of the project is like a waterfall. The central idea behind the Waterfall model is that when you are done with a phase, there is no possibility of returning to it later. Therefore the team needs to make sure that the requirements and the design are correct early on in the project. This can save a lot of time and effort later on [31]. The reason for this is that when you are done with a phase, there is no possibility of returning to it later.

3.8.3 Scrum versus Waterfall

There are several reasons why we chose Scrum over other agile software development processes, and most importantly, why we chose Scrum over the Waterfall method. These points are the most relevant to us regarding the project size, project time and complexity.

- Adaptability – This is one of the most important points. Scrum allows us to be much more reactive and responsive to changes in the project requirements and risks, much thanks to the iterative sprints and what each sprint contains, as explained above. The Waterfall model is strictly sequential. It is impossible to make changes to already committed phases, requirements and similar topics.
- Relationship with customer – In Waterfall, the only place where the customer has any say is in the Requirements phase. There is little or no feedback throughout the project until the final evaluation/verification phase. In Scrum, however, the customer is much more involved with the project team during the development. This is good for both parties. There is continuous feedback and evaluation throughout the project's duration. This will help us avoid bigger mistakes, focus on what is important and deliver the desired product to the customer.
- Estimations and priorities – Our team has limited experience in dealing with estimations in larger projects. By using Scrum, we will have more than one sprint. By the completion of the first sprint, we will have learned more about estimations and the estimations for the next sprint will be improved. This is a good way to gain experience quickly during a project.
- Structure and guidance – The Waterfall model is very vague. What each phase should or should not do is oftentimes unclear. It is also outdated. The Waterfall model has remained more or less unchanged since its inception [23], while software has increased in complexity and time. Scrum, on the other hand, is clearer because it has a set of predefined rules, roles and tasks. This makes each step of the model easier to understand and follow.
- Development – The tangible results are achieved more quickly in Scrum because of the incremental sprints. There is little focus on documentation and more on actual development, contrary to the Waterfall model.

3.9 Development technologies

Java will be our main programming language due to our customer's requirements - integrate Skype. We will be using SQL to communicate with the database.

3.9.1 Java

Java is an object-oriented programming language that has been around since 1995. Since then it has grown to be powerful, yet simple enough to understand. It is widely used today to develop applications of various scales. It will be used to develop our device application. We chose this language because most of the implementation team has had a previous experience with it and Skype has a Java API which will be used for our application. More over, Java has a Graphical User Interface (GUI) framework called Swing which provide a predefined set of GUI components. This framework will be very useful for the implementation of our application.

3.9.2 SQL

SQL is a query language used for managing data in relation to database management systems. It is used for creating, updating and searching data from databases. In our system it will be used to fetch an office owner's data (name, title, contact information, etc.) from a MySQL database.

3.9.3 Eclipse

We decided to use Eclipse Integrated Development Environment (IDE) in our project because a majority of the implementation team has had experience with it from earlier projects. Moreover, Eclipse has a good plug-in called *Subclipse* for Subsversion (SVN) which is very easy to use, even for people who are not familiar with SVN. SVN is a software versioning and a revision control system distributed under a free license. It is used to maintain current and historical versions of files such as source code.

Chapter 4

Requirement specification

This chapter introduces the requirement specifications of our project. It covers the behavioral requirements through use-case descriptions and functional and non-functional requirements through the derivation of several sources included, but not limited to, use-cases, customer meetings and the problem set.

The purpose of introducing this chapter is to give any individual, with or without any technical background, a straightforward explanation of the system's functionality, behavior, overall presentation and desired goals set by the implementation team together with the customer. This will make the design, implementation and testing phases much easier, since we have clearly stated what needs to be done. This reduces any potential ambiguity of the project.

The scope of this chapter and its contents is to sufficiently satisfy our customer's needs and to deliver a software product that meets our goals. We have not added any functionality that was not requested by the customer, as one of our top goals is to create a product that is based on their needs and not on ours. This also gives us extra time and enables us to maintain our focus on the tasks with the highest priority in order to make the software product a success.

4.1 Use-cases

A use-case can consist of several parts, for example textual descriptions and diagrams. Its main purpose is to describe a series of actions that need to be done in order to fulfill a requirement stated in the requirement documentation or to display a desired behavior in a system.

4.1.1 Use-case diagrams and actors

A use-case diagram is a type of behavioral diagram that works as a visual representation of the functionality of the system in the context of an actor. An actor is a user interacting with the system, in whatever way that may be. There can be many different actors in a system, some examples being *End user*, *Owner* and *Student*. These actors each have certain constraints and responsibilities in the system. The actors in our use-cases are *Owner* and *Visitor*.

Actor ID: A1. Owner
Description: This is the owner of the door sign. The door sign is displayed outside the owner's office. The owner will only interact with the door sign through a remote device.
Examples of actions: Has an application with a simple/minimalistic GUI and the ability to write messages and to stream video.

Actor ID: A2. Visitor
Description: The Visitor will only interact through the door sign.
Examples of actions: Viewing information on the door sign, interacting with it and making calls to the owner of the office through the door sign.

This shows that the Owner and Visitor are using two different systems independently that are connect through the Internet.

4.1.2 High-level use-case diagrams

This diagram below displays our desired behavioral requirements for the two actors; Owner and Visitor. The circles represent what kind of functionality the actors can perform, each explained in more details in section 4.1.3 Low-Level Use-Case Diagrams.

4.1.3 Low-level use-cases

This section consists of 8 low-level use-cases, which consist of a diagram and a textual use-case. These low-level use-cases are a derivation of the high-level use-case diagram presented in picture 4.1.

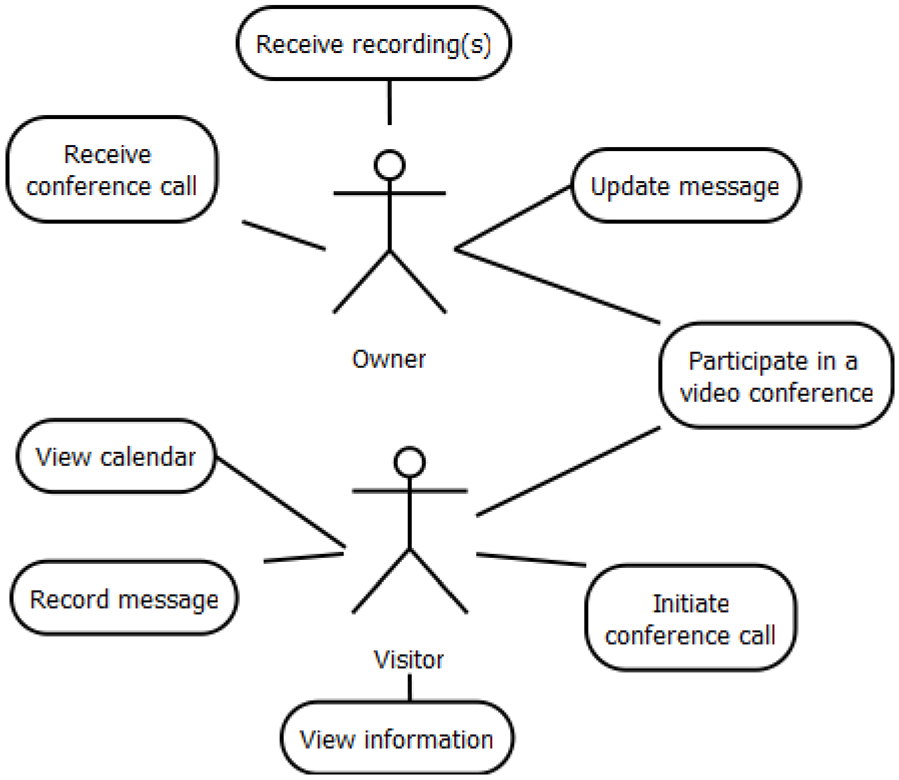
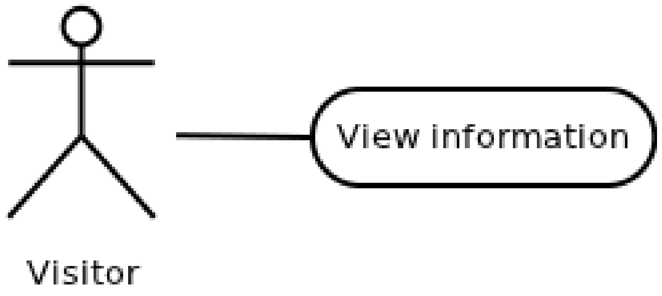
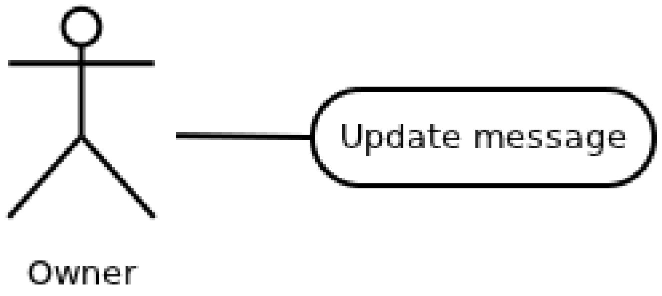


Figure 4.1: High-level use-case diagram



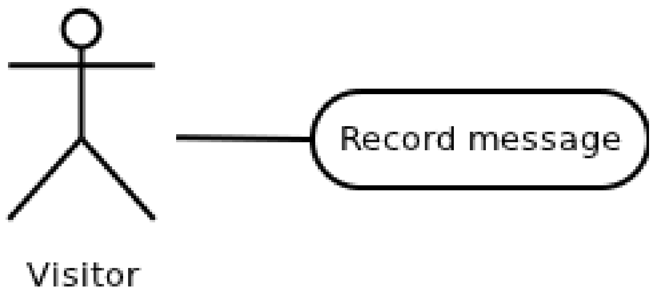
Use case name:	UC1 – View information
Actors related:	Visitor
Trigger:	A person is visiting an office with an interactive door sign
Objective:	To get information about the owner of the office, and see if he/she is available.
Pre-conditions:	The visitor must physically view the door sign.
Post-conditions:	Information about the owner of the office.
Normal event flow:	1. The visitor reads the static information (owner's name, title, room number and similar) 2. The visitor read the personal message.
Variations in the event flow:	2a. There is no personal message for the visitor to read.
Scenario/User story:	As a visitor, I want to have an overview of person working in the office, also I want to be able to read his personal message.

Table 4.1: View information



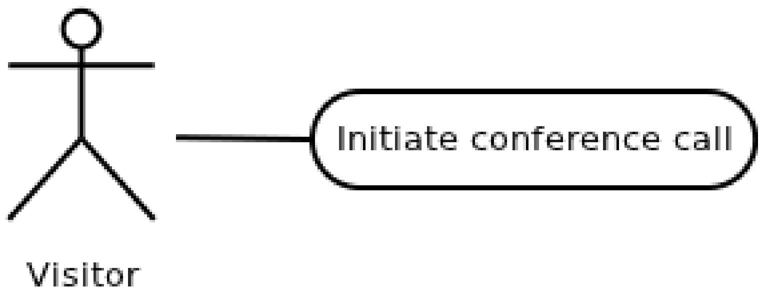
Use case name:	UC2 – Update message
Actors related:	Owner
Trigger:	The owner updates it's personal message through a remote device.
Objective:	To display and update a personal, and customizable, message on the door sign Preconditions: The owner's deceive must be connected to the Internet. An application able to upload the message to the door sign.
Post-conditions:	The personal message on the door sign is updated.
Normal event flow:	1. The owner types in his/hers personal message through an application on the remote device. 2. The message is received, and possibly parsed, by the door sign device's primary application. 3. The message is displayed on the door sign.
Variations in the event flow:	2a. The message is never received, due to packet-drop, parsing error, the Internet connection or something else technical. 2a1. Return to 1.
Scenario/User story:	As an owner of the office, I want to be able to upload a a short message remotely to the door sign with an Wi-Fi/Internet device.

Table 4.2: Update message



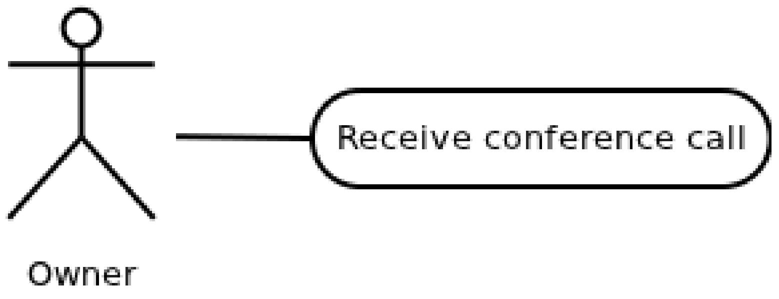
Use case name:	UC3 – Record message
Actors related:	Visitor (Primary) & Owner (Secondary)
Trigger:	The Owner is not available to receive any conference call, and the visitor leaves a recording instead.
Objective:	To record a video/audio message via the door sign, as a voicemail, to the owner.
Preconditions:	The device of the owner, for receiving conference calls, is offline. The visitor must be interact with the door sign. A specific amount of recording time is only available.
Post-conditions:	The recording must be stored on the door sign device, ready to be uploaded to the Owner, per request.
Normal event flow:	1. The Visitor confirms he wants to leave a recording 2. The Visitor interacts with the door sign camera 3. The Visitor gets a confirmation that the recording has been restored.
Variations in the event flow:	2a. There door sign device cannot store more video/audio 2a1. Remove oldest video entry, or more if is not enough. 2a2. Return to 3.
Scenario/User story:	As a visitor, I want to be able to record a video message to the owner of the office via the door sign camera and I want to be able to end the message at my preference. As an owner of the office, I want the recording of a video message to end after a certain amount time, so I does not take too much space and time to upload.

Table 4.3: Record message



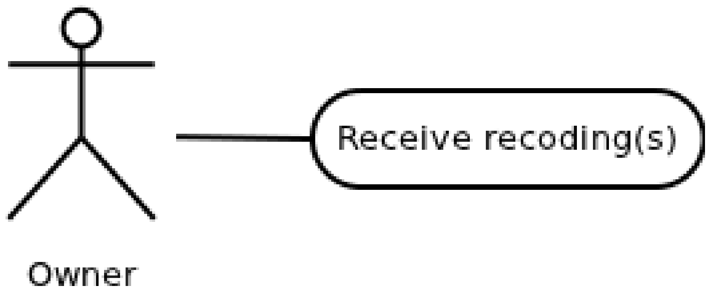
Use case name:	UC4 – Initiate conference call
Actors related:	Visitor
Trigger:	The Visitor wishes to contact the owner.
Objective:	To have a video conference with the Owner of the office via the door sign.
Preconditions:	The Owner of the office's receiving device must be connected to the Internet. The Visitor must interact with the door sign.
Post-conditions:	The owner answers the call or doesn't answer.
Normal event flow:	1. The Visitor initiates the conference call by pressing a "Call/Contact" button on the Door Sign. 2. The Visitor wait for an answer from the Owner.
Variations in the event flow:	2a. The Owner rejects the call 2a1. Return to 1.
Scenario/User story:	As a visitor, I want to be able to initiate a video conference with the owner of the office via the door sign

Table 4.4: Initiate conference call



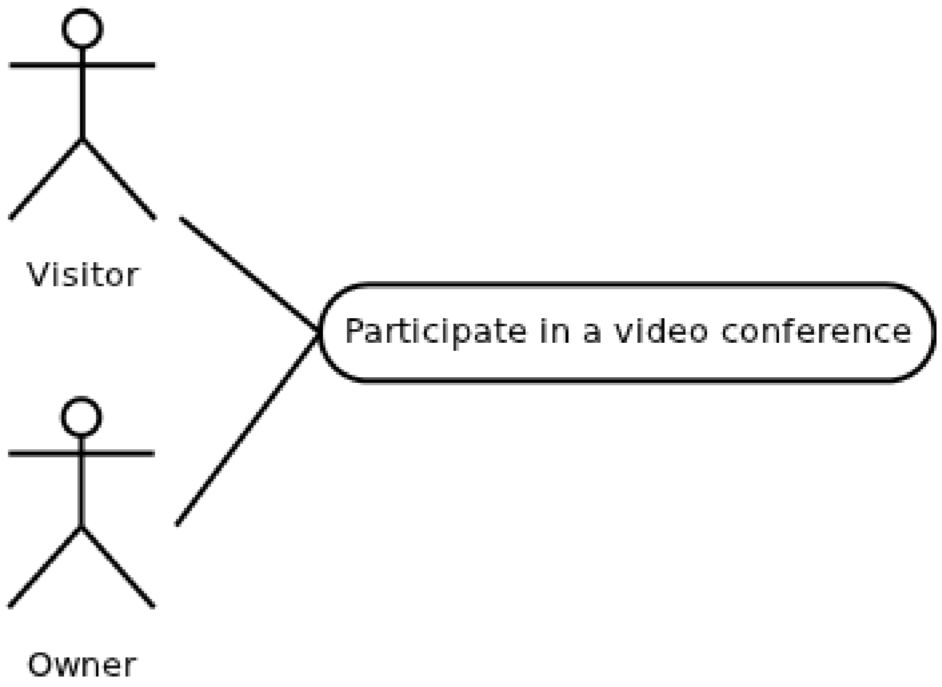
Use case name:	UC5 – Receive conference call
Actors related:	Owner
Trigger:	A Visitor trying to contact the owner through a conference call.
Objective:	To have a video conference with the Visitor through the door sign.
Preconditions:	F3: Initiate conference call
Post-conditions:	N/A
Normal event flow:	1. The Owner receives a call from the Door Sign 2. The Owner chooses to reject or accept the incoming conference call
Variations in the event flow:	2a. The Owner might not have time to reject or accept the incoming call, the Visitor *hangs up*. 2a1. Return to 1.
Scenario/User story:	As an owner of the office, I want to be able to receive any incoming conference calls from visitors when my receiving device is connected to the Internet.

Table 4.5: Receive conference call



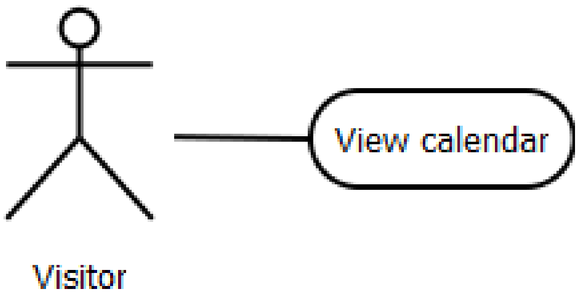
Use case name:	UC6 – Receive recording(s)
Actors related:	Owner
Trigger:	The Owner's receiving device connects to the Internet and one or more unheard recordings are available.
Objective:	To receive any recorded messages left by any visitors during an offline period.
Preconditions:	Owner's receiving device must be online. The Visitor(s) must have left a recorded message during the Owner's receiving device was offline.
Post-conditions:	The ability to stream the recorded message left by the Visitor(s)
Normal event flow:	1. The Owner gets notified that there are recordings left on the Door Sign. 2. The Owner chooses one of the recordings 3. The Owner plays one of the recordings
Variations in the event flow:	3a. It is not possible to play the recording due to a bad Internet connection. 3a1. Return to 2
Scenario/User story:	As an owner of the office, I want to be able to receive and play any recorded messages from visitors that may have happened during the period my receiving device was offline, and additional information like timestamps.

Table 4.6: Receive recording(s)



Use case name:	UC7 – Participate in a video conference
Actors related:	Owner, Visitor
Trigger:	The Owner accepts the incoming call from a Visitor
Objective:	The have an successful video conference.
Preconditions:	F4: Initiate conference call, F5: Receive conference call
Post-conditions:	Satisfying streaming quality achieved.
Normal event flow:	1. The two Actors engage in a conversation
Variations in the event flow:	1a. Due to bad Internet access, it is impossible to have an understandable conversation. 1b. Due to bad streaming protocols, it is impossible to have an understandable conversation.
Scenario/User story:	N/A

Table 4.7: Participate in a video conference



Use case name:	UC8 - View calendar.
Actors related:	Visitor.
Trigger:	The visitor selects the calendar option in the information view.
Objective:	To see when the owner of the office is available and/or unavailable today.
Preconditions:	The Visitor must interact with the door sign.
Post-conditions:	An overview of the workday of the owner.
Normal event flow:	1. The Visitor views to calendar 2. Views to actual timeslots with emphasized colors indecating the availibility.
Variations in the event flow:	
Scenario/User story:	As a Visitor I want to be able to view the owner's current workday, displaying when he is available, in a meeting, working and similar.

Table 4.8: View calendar

4.2 Requirements

In this section the functional and non-functional requirements will be presented with additional information regarding priorities, complexity and more.

Both the functional and non-functional requirements are described in a list, and each list contains the following four columns:

Requirement No. A short and unique notation describing what requirement we are discussing.

Description A description of the functional requirement, stating the most important aspects regarding the functionality the system must fulfill.

Complexity By connecting one or more user stories to the requirement, the project group assigned complexity points through Scrum's planning poker.

Priority The priority of each functionality is assigned to be low, medium or high.

- Low – This is not a vital requirement in the context of succeeding the project goals, rather an option.
- Medium – An important requirement with regards to the overall quality of the software. If implemented, it will greatly enhance the software and project, but the project is not deemed a failure if not implemented.
- High – A requirement with a high priority represents a crucial part of the software. If not implemented, the project will be seen as a failure.

4.2.1 Functional requirements

The functional requirements state what the system must be able to do. This list will satisfy both the use-cases and business requirements that focus on the functionality of the system

4.2.2 Non-functional requirements

Non-functional requirements, also known as system quality attributes, and functional requirements are orthogonal [4]. It describes the desired qualities of how a system should operate and what it should focus on. Examples of non-functional requirement categories are availability, security, performance, modifiability, usability and maintainability.

Requirement No.	**Description**	**Complexity points**	**Priority**
F1	The device should be able to display a picture, the first and last names of the owner, the owner's title, and a short message on its screen.	2	H
F2	It should be possible to update the short messages remotely.	20	H
F3	It should be possible to start recording a short audio and video message. If not stopped manually, the recording will automatically stop.	8	M
F4	A recorded message should be stored locally.	3	M
F5	It should be possible to have a video conference with the owner.	40	M/H
F6	The system should have a database with the information of the people who will be using it.	3	H
F7	When the receiving device (to the owner of the office) has established a connection to the Internet, it should receive all messages sent to him/her during the offline period with additional information, e.g. timestamp.	20	L
F8	A visitor should be able to view a calendar regarding the office owner's workday and indicate via a coloring schema his/hers availability in equally divided timeslots.	8	M

Table 4.9: Functional requirements

Requirement No.	**Description**	**Complexity points**	**Priority**
Q1	You should not be able to exit the DoorSign application and access other parts of the device.	20	M/H
Q2	You should be able to learn how to use the system by yourself in a short time.	13	M/H
Q3	You should be able to have a normal conversation through the video conference.	N/A	M

Table 4.10: Non-functional requirements

4.3 Product backlog

As described in section 3.8.1, the product backlog is a part of the Scrum methodology, which is a complied list describing all potential implementation features of the system, derived from for example from the requirements, as in this case.

Feature No.	**Taks**	**Priority**
1	Displaying Information	H
2	Update message remotely	H
3	Database	H
4	Security Q1: Application restriction	M/H
5	Usability Q2: Designing GUI	M/H
6	Record video	M
7	Store video	M
8	Video conference	M/H
9	Send video messages	L
10	Performance	M
11	Calendar	M

Table 4.11: Product backlog

Chapter 5

Software architecture

In this chapter we will discuss what our architectural drivers are, and how they will affect our overall architecture. We will cover tactics, patterns and views we have chosen to use, which will be rationalized at the end of the chapter.

Attribute driven design was used to create the software architecture. In this section you will find an explanation of the architectural drivers, and the tactics and patterns that will be implemented to satisfy them. There is also a section with different views that explain the architecture in more detail from different perspectives.

5.1 Architectural drivers

The architectural drivers for this application will be the functional requirements F1 and F2, found in table 4.9, and the non-functional requirements Q2 and Q3, found in table 4.10. The functional requirements are concerned with how information is stored and how it is updated. Quality requirement Q2 describes how information is to be displayed visually so that the user interface is easy to learn and understand. Q3 sets certain demands to the performance of the video conference functionality.

5.1.1 Requirement F1

It is important that the application displays the correct information about the owner of the office. All information about the owner will be stored in a database. The information should update automatically if changes are made to the database or if you get a new remote message through the Internet connection.

5.1.2 Requirement F2

One of the main functionalities of this application is to update your status remotely. To do this, the application will need to be able to receive information through an Internet connection, and then update the proper information. We will need a networking part that takes care of all connections to the outside world. This networking part should be separate from the rest of the system to localize changes.

5.1.3 Requirement Q2

The part of the program that has to do with the visual representation of information and it should be separate form the rest of the program. Given our focus on usability, the user interface will probably change many times during the development phase of the project. Therefore, it will be a good idea to maintain the code for the user interface separate from the rest of the system to localize changes.

5.1.4 Requirement Q3

For the video conference functionality to be usable, it is important that the delay of the conversation does not exceed 300 ms. One way to ensure this kind of performance is to use a third party software.

5.2 Architectural tactics

5.2.1 Usability

Design-time tactics

To ensure good usability we will have to make regular changes to the user interface to reflect users' input and to make improvements if our usability test gives bad results. Therefore we will be using the model-view-controller (MVC) pattern witch separates the user interface from the rest of the application.

Runtime Tactics

We will also maintain a model of the system to help us achieve good usability. The user should be able to see what state the system is in at any time and should intuitively understand what actions, if any, he or she can make (for example through progress bars).

The system will also support user initiative in form of buttons to initiate, stop or cancel video-call. With system responding to user actions in form of labels we get a mixed-initiative that requires event listeners and event handlers.

5.2.2 Performance

To solve possible performance issues connected with video conference, we decided to use third party software in form of SkypeKit [10]. SkypeKit provides a runtime that does all the optimization work for us.

Skype has its own optimization techniques and requires as little as 128 kbps connection speed to initiate a low quality video call.

Our only concern is that Skype operates in super node mode, using up bandwidth, thus interfering with connection speed and integrity. Fortunately this feature can be manually disabled.

System parts not including Skype consist of a small Java application that will hardly use any resources at all.

5.3 Architectural patterns

5.3.1 Model-View-Controller

Since high usability of our user interface is one of our architectural drivers, separating the user interface from the rest of the application will help to localize changes. To achieve this in practice, the model-view-controller pattern will be used in this application. This architectural pattern separates the model, containing the information to be displayed from the controller, which updates the model, and the view, the graphical user interface. In general it works like this:

1. Someone interacts with the GUI

2. The controller interprets the interaction in a way the model can understand
3. The controller notifies the model of the interaction
4. The model updates
5. The view, listening to changes made to the model, updates as soon as the model propagates the changes

Java has a good support for the model-view controller pattern through the java.beans library. Therefore it should not be much of a challenge to implement.

5.3.2 Event-Driven Architecture

With this pattern we can create one-to-many dependency between objects, so that when one object updates, this change will be propagated to all the ones that are listening. This will be used in the MVC-structure as well as for buttons on the screen or device. Creating events and listeners in Java is well supported through the standard libraries. There should be no problems in implementing this functionality.

5.4 Design patterns

5.4.1 Singleton

The singleton pattern is used to ensure that only one object is allowed to be initiated from a specific class.

Very intuitive and easy to implement.

5.5 Architectural views

We decided to use 4+1 architectural view model, because it suited our approach of designing our architecture. Using this approach we decomposed our system into 4 views. These views are targeted at different stakeholders, to make the system easier to understand (for the customer and users) and to give the development team overview of what needs to be done and how the system is structured.

To make system comprehensible for the end-users we used use-case view. This view consists of use-case diagrams with short description.

Logical and Development view are made for the implementation team. These views describe structure of the system and relationship between system components.

Process view consists of activity diagrams, that show possible scenarios of how the system can be used. These diagrams are also useful for the implementation team to get an overview of the system run-time processes and do some brainstorming on how to implement these processes. This view is also usable for the end-user to see all possible actions and their results.

Physical view describes how our system is connected to various hardware components. This view provides high-level overview of the system with short description of components that system is connected to.

5.5.1 Logical view

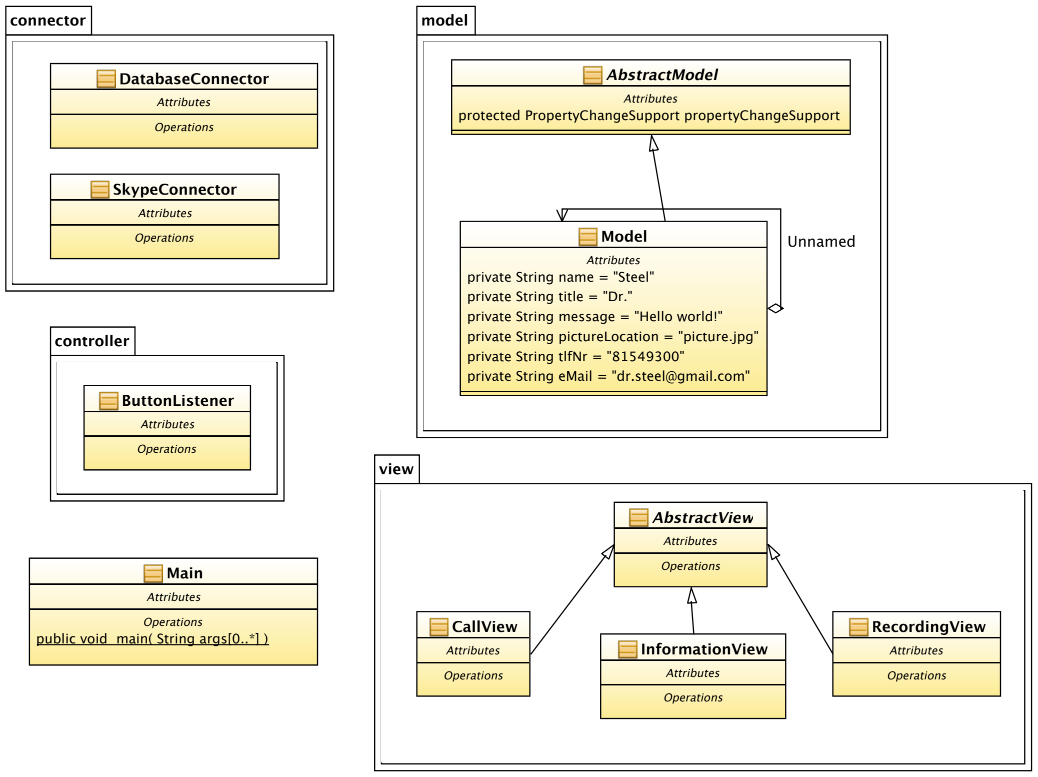


Figure 5.1: Class diagram

Class diagram description

Figure 5.1 is a a picture of our class diagram. Here is a short description of the classes of each package.

- **root-folder** – The Main class contains the main-method that is called when the program is started.
- **model** – Contains two classes, an *AbstractModel* and a *Model*-class. The *Model* class will extend *AbstractModel* and contain all the information to be displayed on the different views. *AbstractModel* uses *java.beans.PropertyChangeSupport* to propagate changes made to the model.
- **controller** – Only contains *ButtonListener* which will detect if the user is pressing buttons on the screen, and take proper action.

- **view** – Contains all the views we will be using. The *InformationView* will be the default view and will display information about the owner of the office. *RecordingView* is used when you record
- **connector** – This package handles all connections. *SkypeConnector* takes care of all communication with skype and the *DatabaseConnector* gives us access to the database containing name, title, picture etc.

Communication view description

A few communication diagram were made to show how the system responds to certain events initiated by the user or the system.

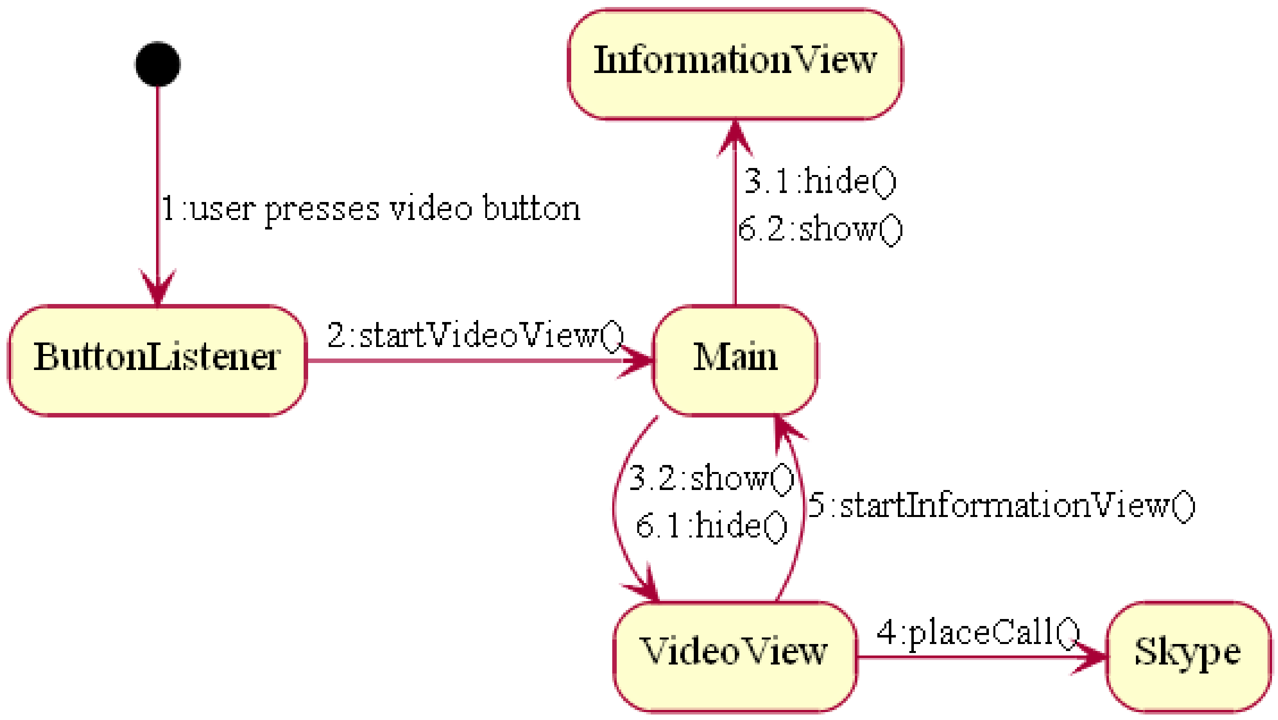


Figure 5.2: Communication diagram – Video conference

The first communication diagram, figure 5.2, shows what happens when the user presses the video conference button:

1. User interacts with the device by pressing the video conf. button.
2. The *ButtonListener* receives the button event and invoking startVideoView() in the *Main* class.
3. When this method is called the *Main* class hides the current view and shows the *VideoView.*
4. *VideoView* then tires to start a video conference by invoking placeCall() in *SkypeConnector.*
5. When the conference is over the *VideoView* invokes startInformationView().
6. The *Main* class then switches back to the default view, which is the *InformationView.*

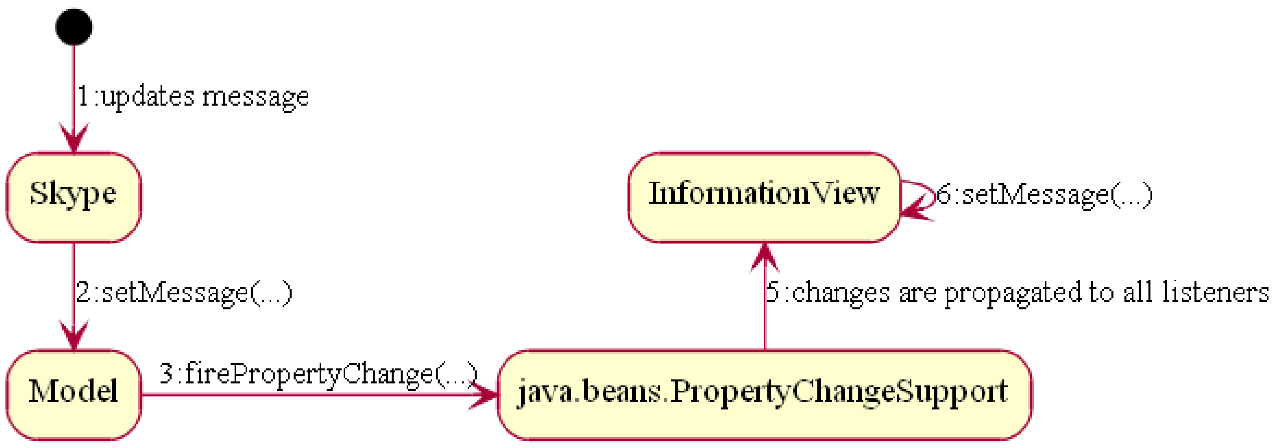

1:updates message
Skype
2:setMessage(...)
Model
3:firePropertyChange(...)
java.beans.PropertyChangeSupport
5:changes are propagated to all listeners
InformationView
6:setMessage(...)

Figure 5.3: Communication diagram – Update message

The second diagram, figure 5.3 shows how the program responds when the message is updated through Skype.

1. A message is received by Skype.
2. *SkypeConnector* invokes setMessage(...) in *Model*.
3. When the new value is set, *Model* uses the method firePropertyChange(...) in the *java.beans.PropertyChangeSupport* class.
4. The change is propagated to all listeners.
5. *InformationView* that is listening to changes made to *Model* notices the change and updates the correct value.

The last communication diagram, figure 5.4, shows what happens when the program starts.

1. Program starts
2. *Main* class initiates all *Views* (*InformationView*, *VideoView*, *RecordingView*, etc.).
3. *Main* class connects to Skype by invoking login(...) in *SkypeConnector*.
4. *Main* fetches information about the owner of the office from the database.
5. Then *Main* sets the correct values in the *Model*.

5.5.2 Development view

The package diagram, figure 5.5, is very simple as the scale of the system is quite small. The arrows explain the association between the different packages.

5.5.3 Physical view

Here is a short description of the deplyment diagram shown in figure 5.6.

- Laptop – Laptop represents a pc/smartphone with Skype application installed on it. Through this application, office owner gets some control over Office client application.

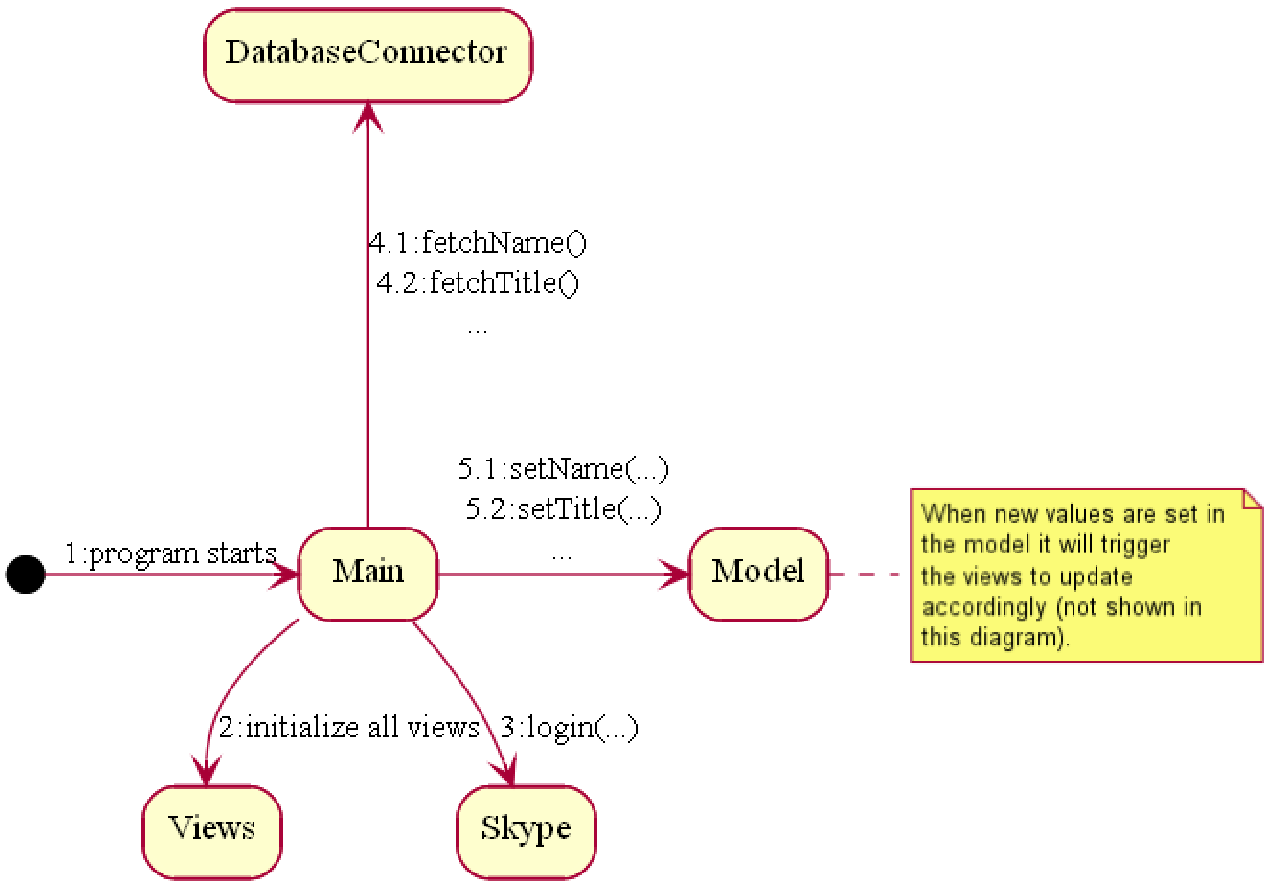


Figure 5.4: Communication diagram – Startup

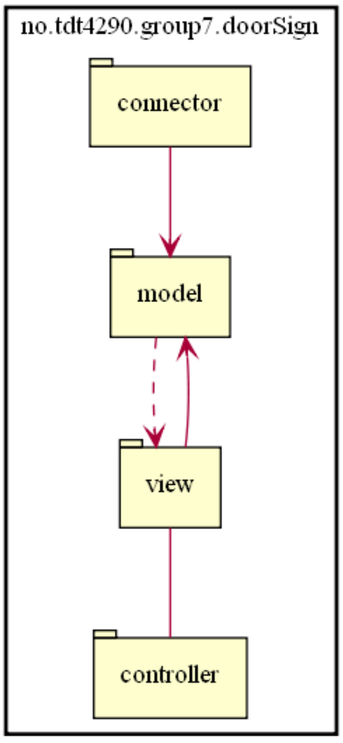


Figure 5.5: Package diagram

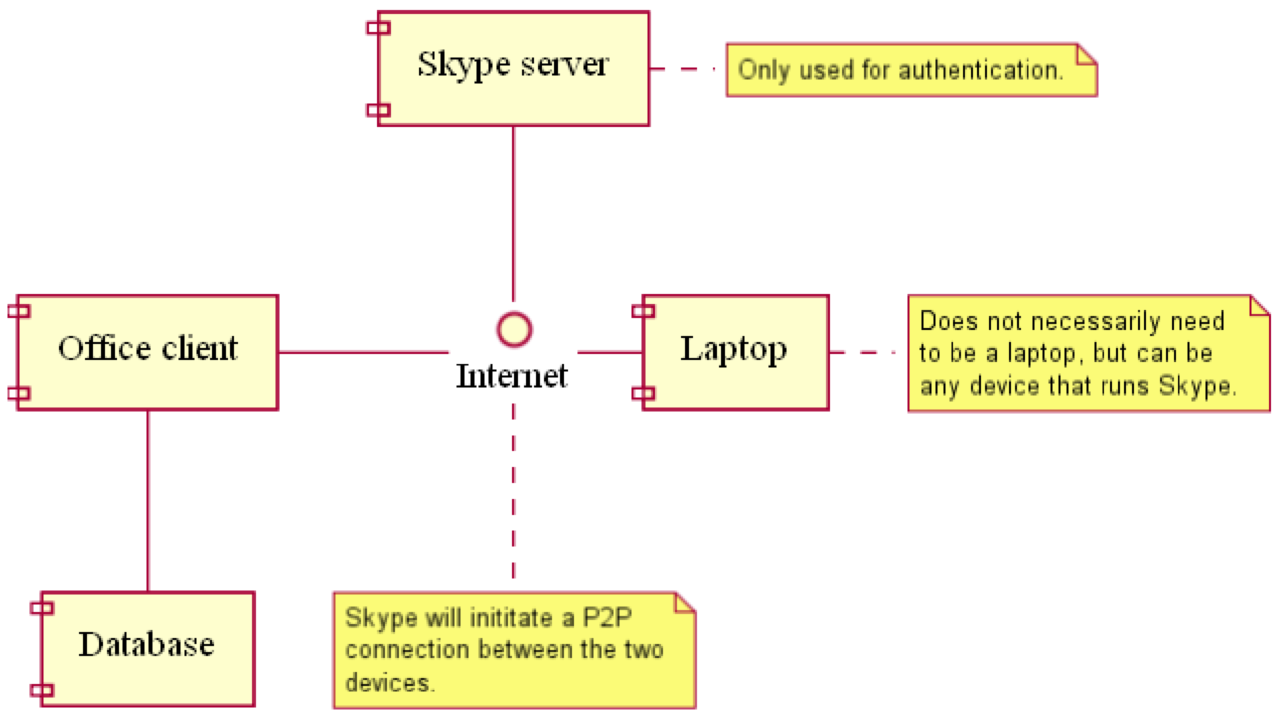

Skype server
Only used for authentication.
Office client
Internet
Laptop
Does not necessarily need to be a laptop, but can be any device that runs Skype.
Database
Skype will inititate a P2P connection between the two devices.

Figure 5.6: Deployment diagram

For example, office owner can update the message simply by sending an instant message to Officeclient as to any other person in his contact list. Through this client office owner will also be able to receive video calls and recorded messages from Office Client users.

- Office client – Office client consists of our application and integrated Skype run-time components to communicate with Laptop. The application will get it's primary information about office owner from the Data Base each time it starts. Integrated Skype components will allow users communicate to office owner using video conference or by leaving a recorded video message. Below is a diagram, figure 5.7, showing how Skype is integrated into our application.
- Database – MySQL database will be running on a remote server. This database will keep information about all office owners, such as: office owners picture, name, title, etc.
- Skype – Skype is a P2P application, that allows users to interact with each other. Skype supports instant messaging, voice calls, video calls, file sharing and much more. Since Skype works as a P2P application it does not have a server through which users communicate, instead users communicate to each other directly or by using *supernodes* – user clients with public IP's acting as directory service that form Skype infrastructure. There is a server that contains all users information but is used for authentication purposes only. To maintain stable, fast, good quality connection, Skype is breaking down sent information into small packages before sending them through Internet.

This top-level block diagram, figure 5.8, shows overall overview of our architecture. The WM8650 device interacts with the user via touch screen, microphone and a speaker. The device connects to the office users pc via USB 2.0. Our system consists of custom made application with integrated SkypeKit. SkypeKit connects to the Skype infrastructure that has P2P architecture consisting of authentication server and SuperNodes to direct traffic. The system gets office owners information from the MySQL database located on a remote server.

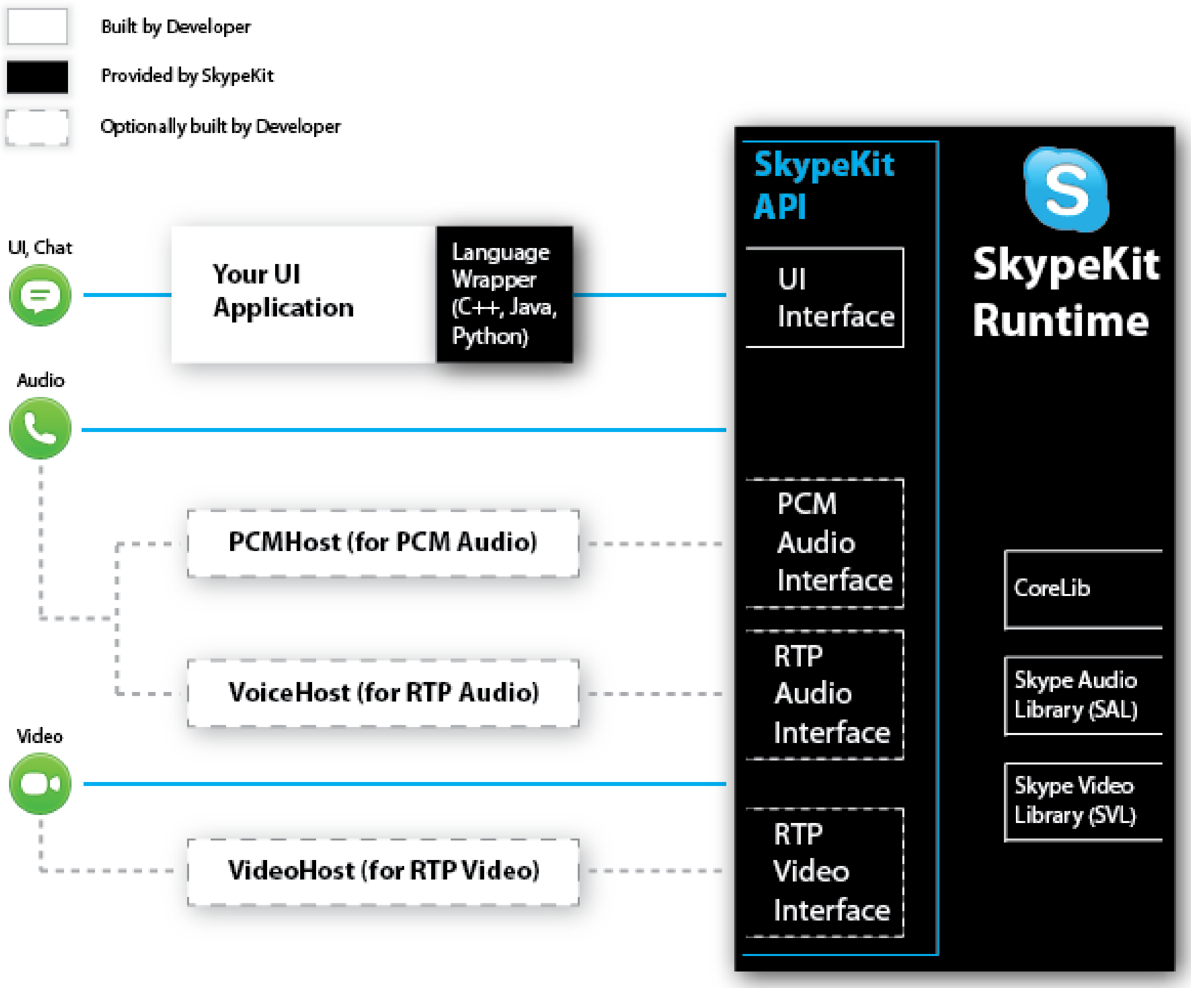


Figure 5.7: SkypeKit integration overview [9]

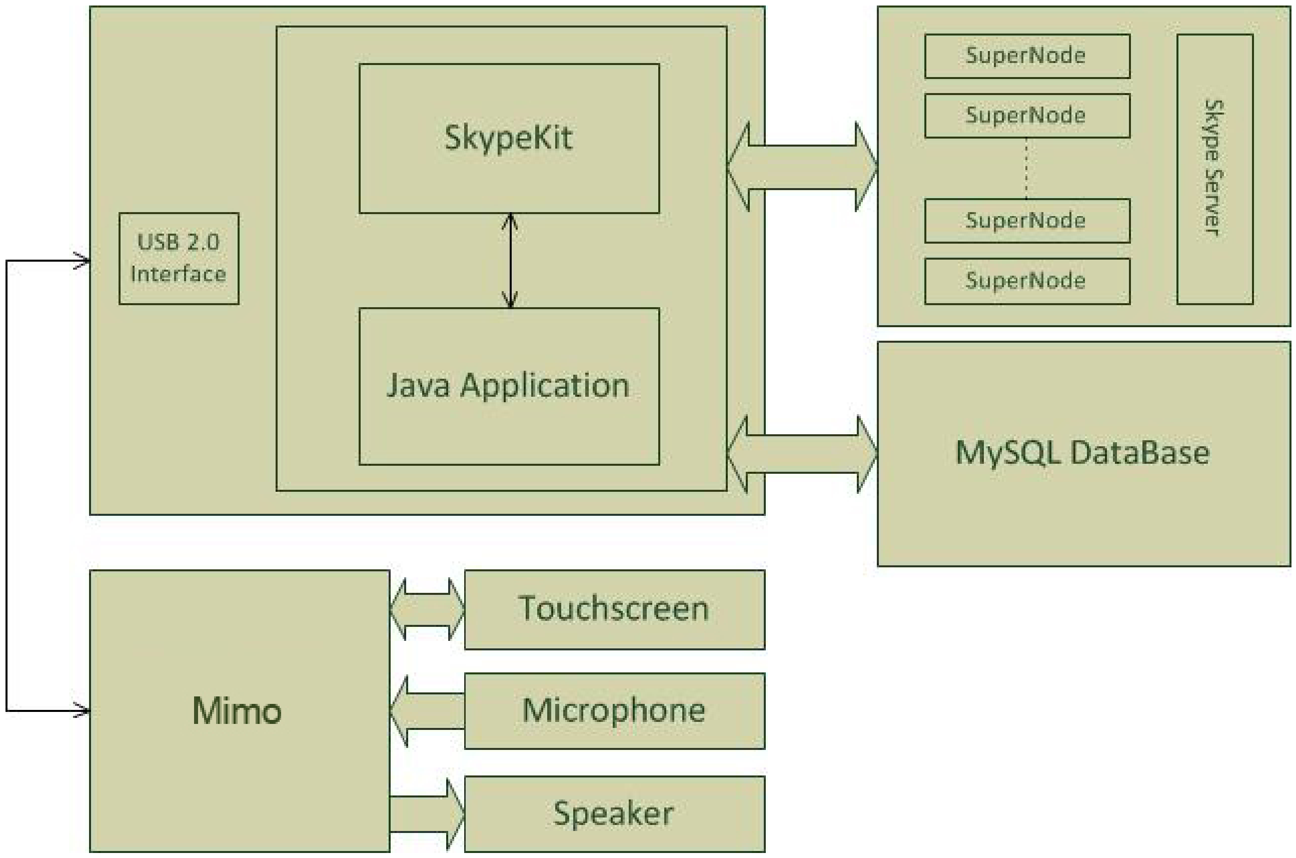


Figure 5.8: Block diagram

5.5.4 Process view

The activities diagrams in figures 5.9 and 5.10, focus on the dynamic aspects of the doorSign.

The figure 5.9 presents the different activities when a visitor wants to call the owner of the office.
Once the call is initiated, three events are possible:

1. The user wants to cancel the call. Then, the call is simply ended.
2. The owner answer the call. Then, the video conference starts. It could stop at any time as soon as one of the two hang up.
3. The owner does not answer. Then, the call is ended after warning the user.

Whichever event occurs, the result is always the same: the call is ended.

The figure 5.10 presents the different activities of the system when the user starts recording a message.
Once the recording starts, three events are possible:

1. The user wants to cancel the recording. Then, the recording is deleted (and not sent).
2. The user manually stops the recording. Then, the recording is stopped and sent to the owner of the office.
3. The timer runs out. As in the previous event, the recording is stopped and sent to the owner.

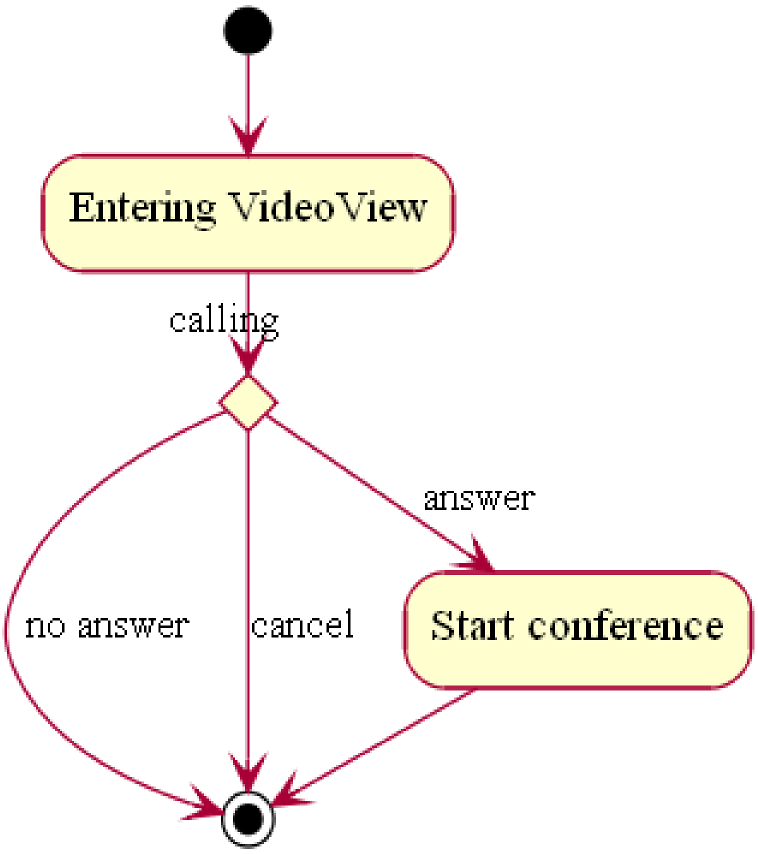


Figure 5.9: Activity diagram 1 – Calling

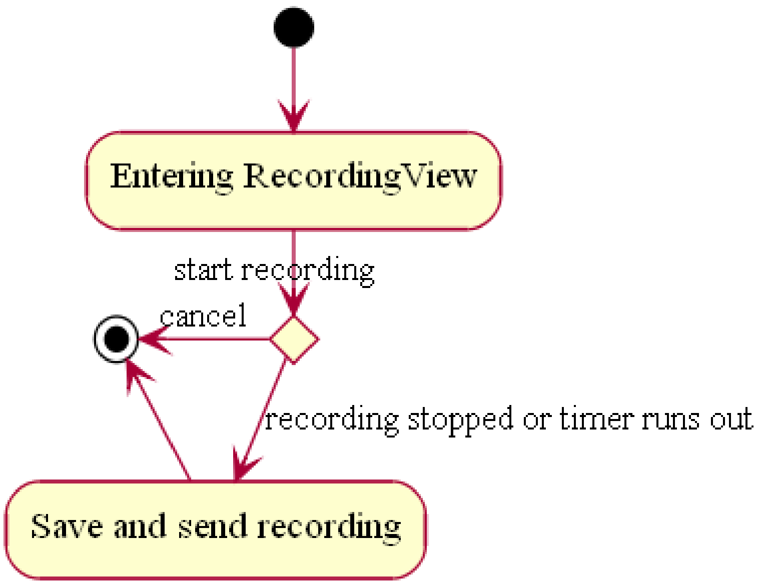


Figure 5.10: Activity diagram 2 – Starting recording

At the end of these three events, the recording is stopped.

5.6 Architectural rationale

We chose to use the Model View Controller (MVC) arcitectrual patter, it was the best way to ensure the creation of a good graphical user interface. A GUI often needs to be changed throughout development process to suit a wide variety of users, so it was important for us to separate it from the rest of the code.

The Singleton pattern is used more for aesthetics rather for its functionality, to avoid using global variables, thus saving some resources. We will also use it to insure that only one object of a class is initialized to avoid complications.

5.7 Resulting architetural changes

This section will present the changes that have been made to the architecture during the implementation phase. Most of these changes were made to the logical and communication view, and the overall logic described in the physical view is still intact. Therefore the overall structure of the architecture has been consistan throughout the whole implementation process, we've successfully avoided any major architectural drift and/or architectural erosion.

5.7.1 Realized logical view

Class Diagrams

Below is the description of changes made to the class diagram during implementation.

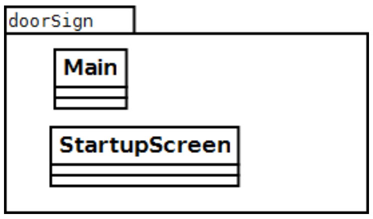


Figure 5.11: Root classes

- **root-folder**[5.11] – The *StartupScreen* is used for user input to select the appropriate video output device.

- **model**[5.12] – This package has three new classes:
  - *Message* – Container class responsible for representing the message received from the office owner through Skype.
  - *VideoMessage* – Container class responsible for keeping and representing the video message, recorded by the device.
- **controller**[5.13] – recoridngHandlers-package was added. This package is responsible for handling the video and audio data received by the application from the device.
  - *VideoAndAudioSender* is responsible for invoking *VideoAndAudioMerger* and sending the file to the office owners Skype account.

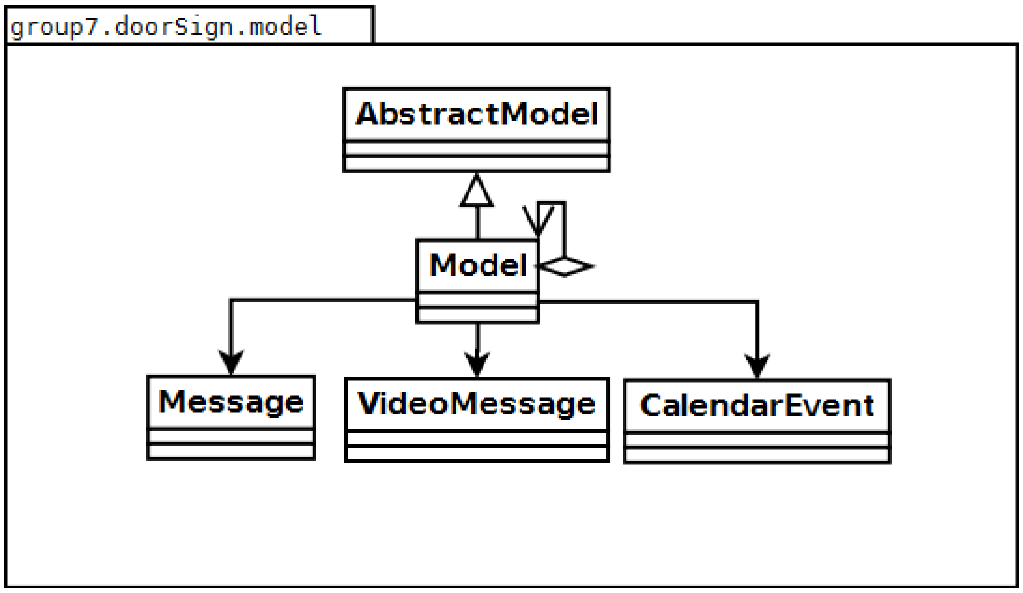


Figure 5.12: The model package

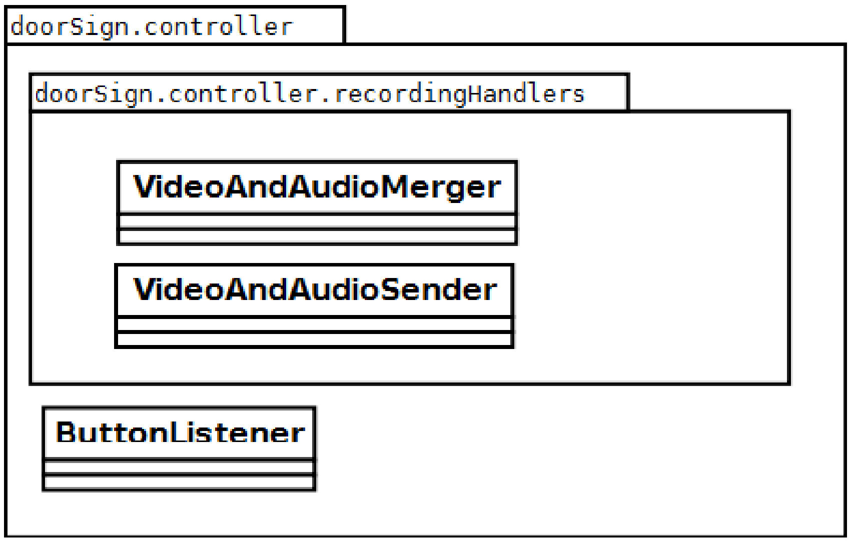


Figure 5.13: The controller package

  - *VideoAndAudioMerger* handles the merging of audio and video files to one video recorded message.

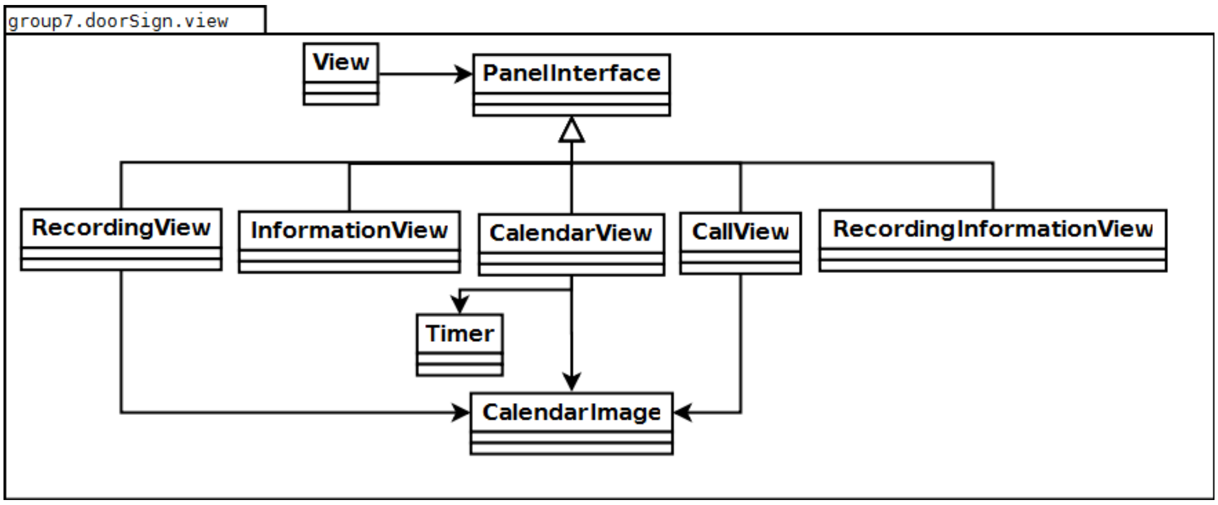


Figure 5.14: The view package

- **view**[5.14] – Contains all the views we will be using. Addidional views created:
  - *RecordingInformationView* was created to give office visitors some feedback on failed calls(if the office owner did not answer a call).
  - *CalendarView* is used to inform the office visitors about office owners current days schedule.
  - *CalendarImage* is a container class, used to store and maintain (as in repaint) the calendar image.
  - *View* maintains what views to show on the screen and the logic for changeing between them.

- **connector**[5.15] – This package was expanded and now contains these classes:
  - *AudioConnector* class, responsible for handling all things concerning audio recording, converting and storing
  - *CalendarConnector* class, responsible for handling the connection to GoogleCalendars and fetching data from it.
  - **skype** package contains:
    * *KeyManager* is responsible for managing keys used to connect to the Skype runtime.
    * *LoginLogoutManager* manages loging in and out of the Skype account.
    * *MySkype* detects personal hardware settings of the machine running the application and set up Skype to use the audio and video devices properly.
    * *SessionListener* – is used to listen for Skype events during at runtime.
    * *SkypeMessageParser* parses the office owners messages.

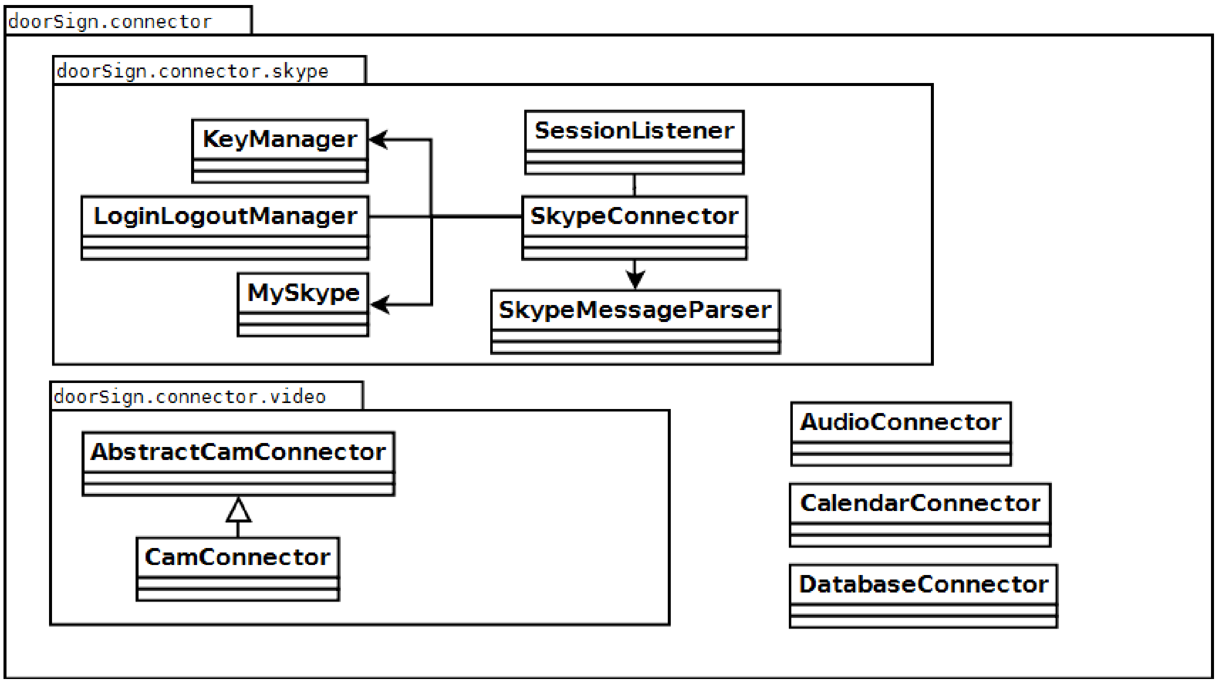


Figure 5.15: The connector package

- **video** package contains
  - *CamConnector* handles the connection to the internal web-cam of the Mimo monitor.
  - *AbstractCamConnector* The *CamConnector* will extend this class.

Communication Diagrams

The first communication diagram, figure 5.16, shows what happens when the user presses the Call button:

1. User interacts with the device by pressing the Call button.
2. The ButtonListener receives the button event and invoking setActivePanel(”cal”) in the *View* class.
3. When this method is called the *View* class removes the current view and shows the *CallView*.
4. *CallView* then tires to start an audio call by invoking placeCall() in *SkypeConnector*.
5. When the call is over the *CallView* invokes setActivePanel(”inf”) in *View*.
6. The *View* class then switches back to the default view, which is the *InformationView*.

The second communication diagram, figure 5.17, shows what happens when the office owner updates his status message:

1. *SessionListener* catches a new event – reception of a new message from the office owner.
2. A message is received by *SkypeConnector*.
3. *SkypeConnector* then it invokes setMessage(...) in *Model*.

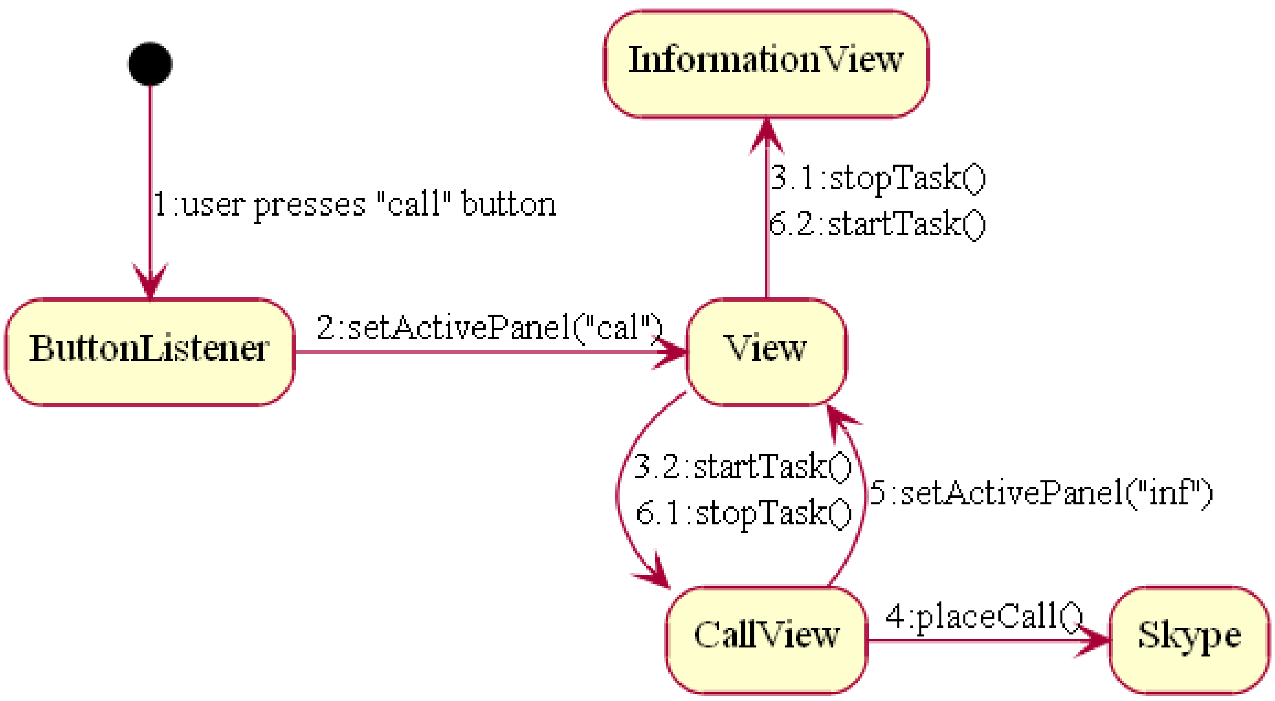


Figure 5.16: Audio call communication diagram

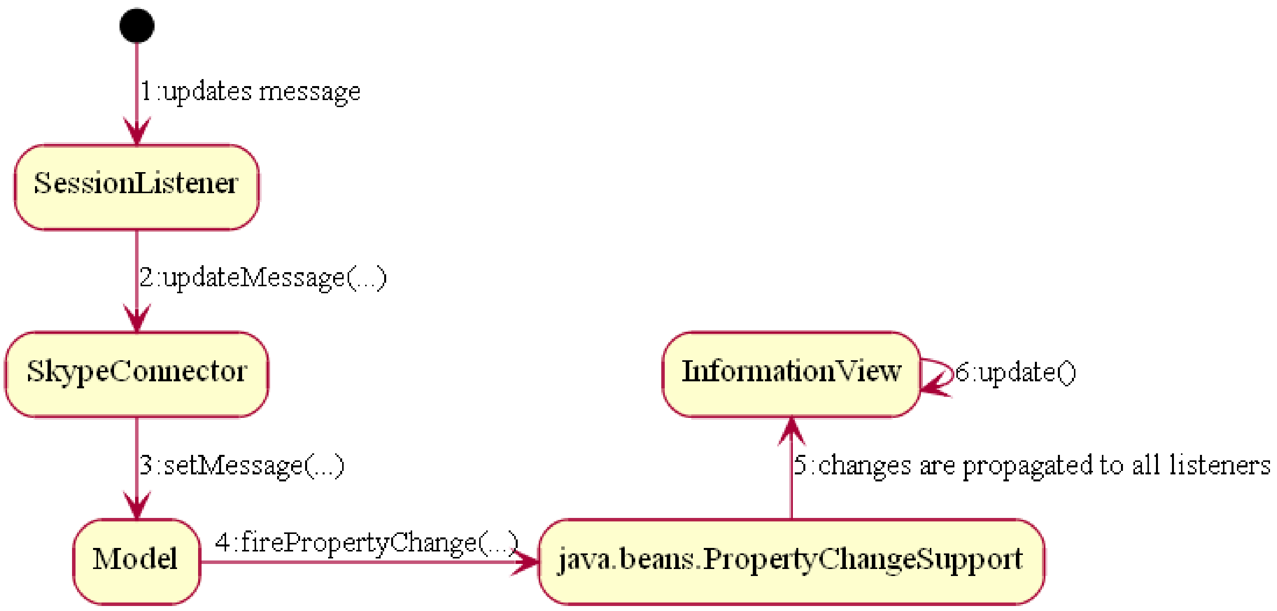


Figure 5.17: Message update communication diagram

4. When the new value is set, *Model* uses the method firePropertyChange(...) in the *java.beans.PropertyChangeSupport* class.
5. The change is propagated to all listeners.
6. *InformationView* that is listening to changes made to *Model* notices the change and updates the correct variable.

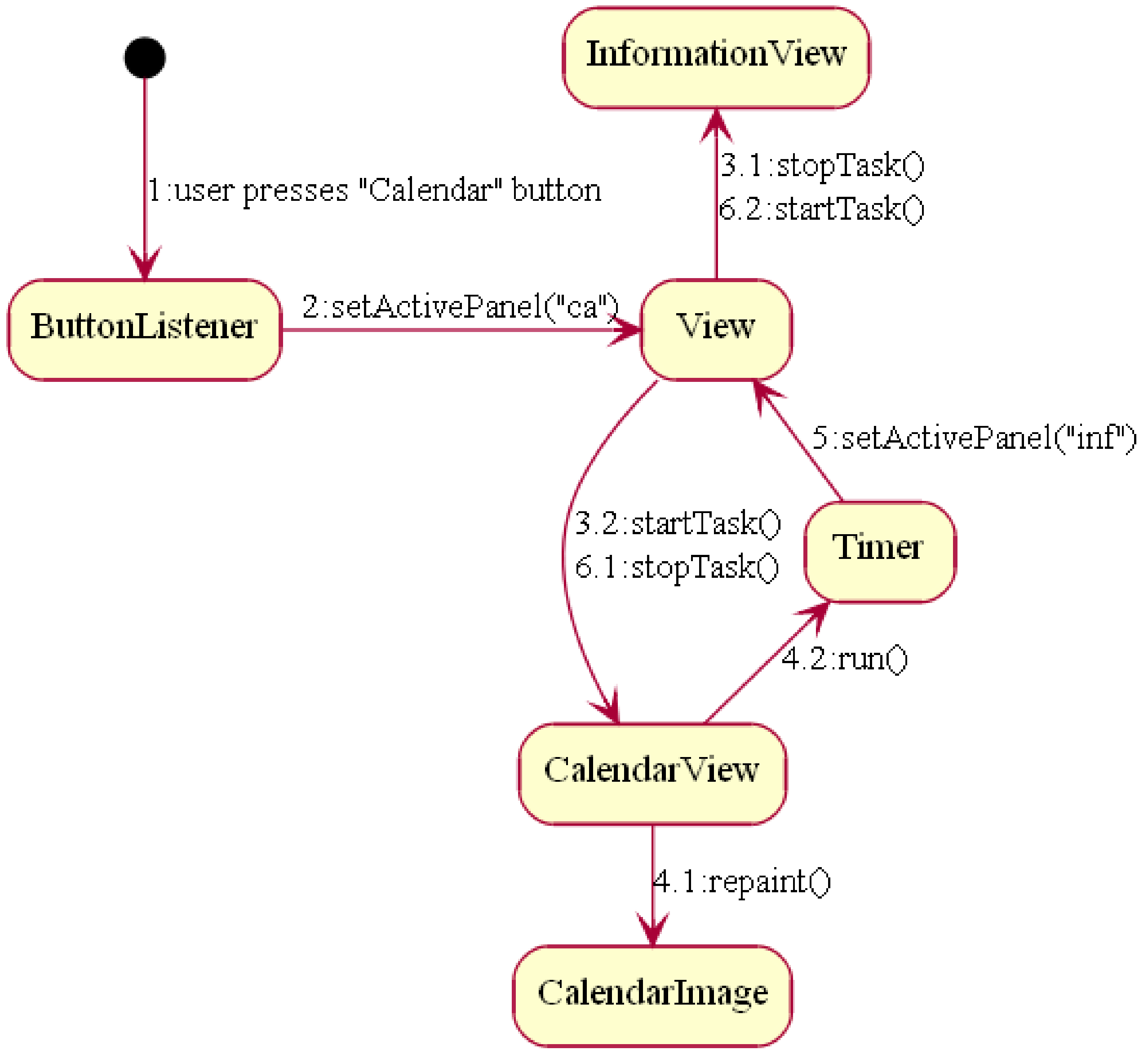


Figure 5.18: Calendar communication diagram

The third communication diagram, figure 5.18, shows what happens when the user presses the Calendar button:

1. User interacts with the device by pressing the Calendar button.
2. The *ButtonListener* receives the button event and invokes setActivePanel(”ca”) in the *View* class.
3. When this method is called the *View* class removes the current view and shows the *CalendarView*.

4. *CalendarView* then tires to show the office owners schedule by invoking repaint() in *CalendarImage*. The method repaint() is called at all times continuesly while the *CalendarView* is shown, to display real time information.
5. When the *Timer* runs out or the user presses cancel, thr *CalendarView* invokes setActivePanel("inf") in *Model*.
6. The *View* class then switches back to the default view, which is the *InformationView*.

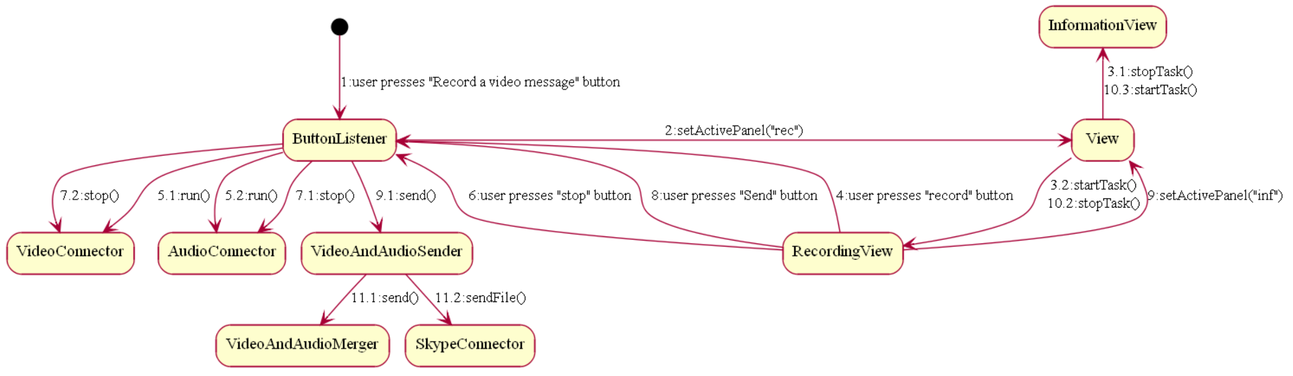


Figure 5.19: Record video message communication diagram

The forth communication diagram, figure 5.19, shows what happens when the user presses the Record Message button:

1. User interacts with the device by pressing the Record Message button.
2. The *ButtonListener* receives the button event and invokes setActivePanel("rec") in the *View* class after that the application is waiting for user input.
3. The *ButtonListener* receives the button event and invokes run() method in *VideoConnector* and *AudioConnector* and the recording of the message starts. After that the application is waiting for user input or for the timer to run out.
4. The *ButtonListener* receives the button event and invokes stop() method in *VideoConnector* and *AudioConnector* and the recording of the message stops. After that the application is waiting for user to press send button.
5. The *ButtonListener* receives the button event and invokes send() method in *VideoAndAudioSender* that start the process of sending the recording to office owner.
6. While sending the video file to office owner, the *RecordingView* invokes setActivePanel("inf") in *View*.
7. The *View* class then switches back to the default view, which is the *InformationView*.
8. In the mean time the *VideoAndAudioSender* invokes send() method in *VideoAndAudioMerger* that will merge the audio and video files to one video file.
9. When the mergind is done, *VideoAndAudioSender* invokes sendFile() in *SkypeConnector* that will result in sending the video file to the office owner through Skype.

The last communication diagram, figure 5.20, shows what happens when the program starts. The only change made to this diagram is that information from the database will be fetched by one method call to *DatabaseConnector* instead of multiple method calls.

1. Program starts

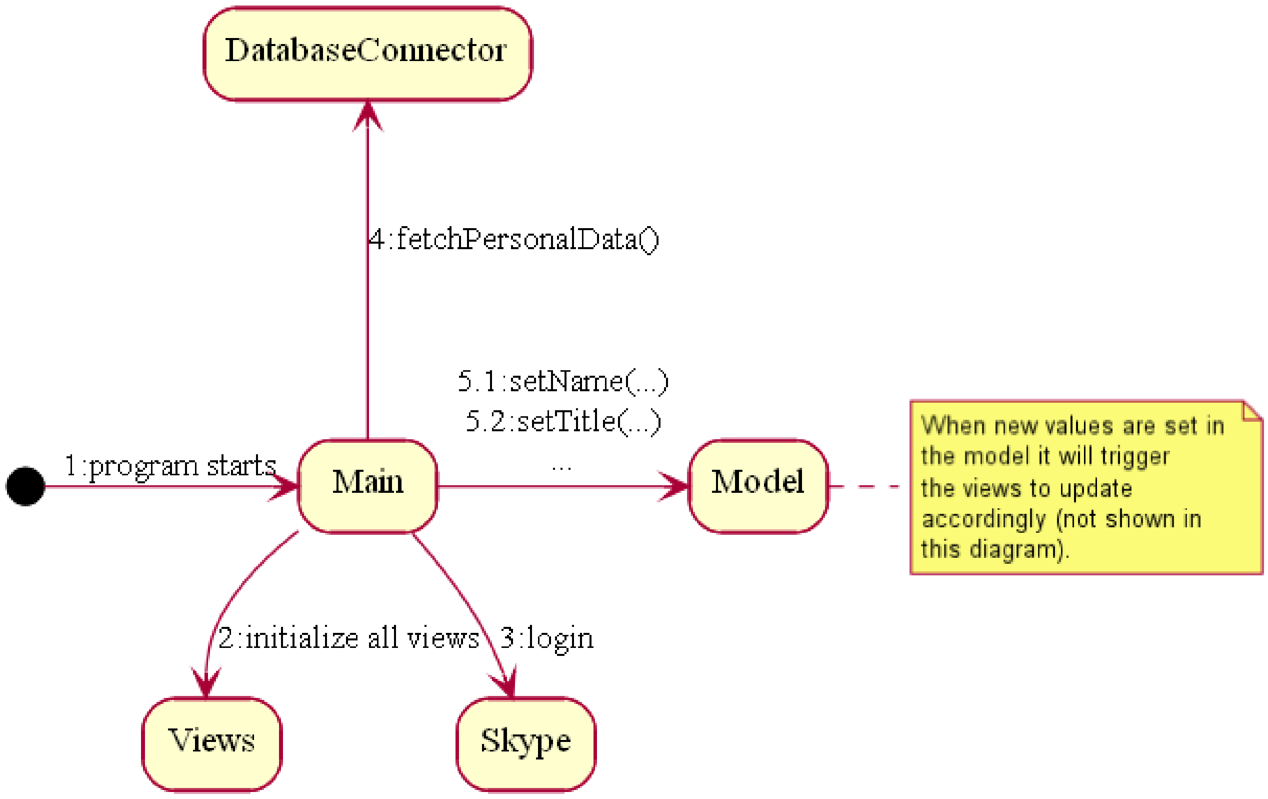


Figure 5.20: Startup communication diagram

2. *Main* class initiates all views (*InformationView*, *VideoView*, *RecordingView*, etc.).
3. *Main* class connects to Skype by calling login(...) in *SkypeConnector*.
4. *Main* fetches information about the owner of the office from the database.
5. Then *Main* sets the correct values in the *Model*.

Chapter 6

Quality assurance

Every project needs quality assurance to ensure an overall quality of the project or process. We have our own definitions of quality, and will therefore strive to uphold the standards we have created to maintain consistency in our documentation and our process in general.

Our customer's happiness is as crucial in this project as it is in any other project in the working world. Customers have their own definition of quality, and their happiness depends on the quality of the product. Therefore, it is natural that we have some level of quality assurance so we can create the kind of system that the customer really wants. To do things in an easier and more efficient way, we need to follow certain standards to maintain consistency in our documentation and our process in general. This allows us to create the best version possible of our product so that our customer is satisfied.

6.1 Communication

6.1.1 External communication

The customer
An important thing to note is that all features and requirements presented at customer meetings must be implemented in order to give the impression of a completed product. Promises we have made must be kept to preserve the credibility of our team.

A lack of communication with the customer could result in the team creating a partially working prototype that only meets a few of the customer's requirements. This could give undesirable consequences, such as losing the customer and thus losing money, gaining a bad reputation and not getting future projects.

It is therefore vital to communicate with the customer to let them know early on if their requirements do not have a possibility of being correctly implemented, or if their requests are unrealistic. Only then will we have the chance to negotiate and come up with new ways and requirements that will satisfy our customer.

To make it easier for the customer to reach out to us, and for us to keep a steady flow of communication with them without stepping on each other's toes, we have appointed a customer contact. The customer contact will maintain communication with the customer and provide them with information on how we are progressing and what could potentially cause problems in the immediate future. That way, if any issues arise, the customer will be aware of them and we will be able to find solutions without having wasted too much time. This will also allow the customer to steer the project in a more preferable direction, giving them a sense of control and involvement in the whole process.

The advisor
We will have weekly meetings with our advisor where we can discuss our progress and the project in general. The advisor will be able to give us important feedback, so maintaining communication with him will be very useful for our project. With valuable feedback, our advisor is making sure our report is as good as possible, and that we are progressing nicely. We have the option to send him e-mails if there are things we would like to ask or discuss. The communication is otherwise limited to the weekly meetings.

By creating weekly reports for him, we are able to steer the conversation in any direction of our choice. It is therefore important that we create good agendas before each meeting so that we can get as much feedback as possible, as this will be necessary for the overall quality of our project.

Having an advisor contact is helpful because the advisor will have a specific team member to contact. There will be little to no confusion. This person will then relay the message to the other team members.

6.1.2 Internal communication

To be able to collectively create a coherent product, we will need to follow set standards for communication in general, implementation work and documentation. A short description of these standards can be found in section 2.7.2 in our Planning chapter.

Communication within the team

The basic foundation was set early on when we wrote our ground rules. If these rules are broken, it could result in disruptions in work flow, a failure to communicate, a lack of motivation in one or more team members and a lack of progress of the project. To prevent this from happening, everyone will have to get familiar with these rules and do what they can to uphold them.

Communicating through e-mail is easy, but to make sure every team member knows what is going on at all times, it is important to use the *reply to all* feature when you reply to an e-mail sent to all the team members. Otherwise information of significant importance may be lost.

To ensure that all team members stay up to date, summaries must be written after each meeting. Team members are required to read the summaries and can also add to them if needed. The templates for these summaries will be discussed later in this chapter.

Implementation

We will have a majority of our team working on the implementation of the different requirements. When a piece of code is done, other team members, as well as future developers and maintainers, should be able to understand perfectly what it does. To guarantee this, we will need coding standards.

As mentioned in section 2.7.2, all class names, methods and variables should be written in English so that they are easily understandable for anyone who might look at the code at a later time. Methods, classes and variables should be described using Javadoc. Formal descriptions explain exactly what something is meant to do and will make the code maintainable on a broader level. Our programmers are coordinating their work using Eclipse, and to prevent major problems, all code should be formatted and all errors must be corrected by the individual programmer before the code is committed.

Documentation

We also use Dropbox as a tool to share various files. To prevent miscommunication and confusion, all files in our folder should have English names that reflect the files' content. If the file name is not descriptive, we will have problems getting a full overview of which files we have and where they belong in the final document. With correctly named files we do not have to open each document to find out what it contains, we can simply look at the file name. This also makes it easier for us to see what still needs to be done, as documents can not be hidden in any way.

As well as being descriptive, the file names should only contain lower case letters and numbers. For spacing between words and numbers, we will use underscores to prevent any errors associated with empty spaces in file names.

As the project grows, the number of files grows with it. Files that belong in the same category should be placed in the same folder so as to make navigation easier. This will help us organize our documents so we can find what we are looking for without wasting time.

References and quotations will be generated using BibTeX. This tool separates different types of references, such as articles, books and web pages and automates the whole process for the writers of the document. This prevents bad or incorrect references, which is especially useful when you have multiple individuals working on the same set of documents.

6.2 Templates

To make the writing process faster, we have created templates for writing summaries of meetings. We have three kinds of meetings; customer meetings, advisor meetings and group meetings. Another template we need is for the weekly report – a document we need to send our advisor every week. These templates can be found in Appendix B, and show the general structure of our summaries and the weekly report.

The date, time (duration) and place for the meeting must always be documented, as well as who attended the meeting. Then follows a short summary in the form of short sentences or full text, documenting decisions and explaining in detail what was discussed during the meeting. The time and place for the next meeting will be found at the end of the document.

6.2.1 Group meetings

Group meetings slightly differ from advisor and customer meetings. We have one full group meeting every Tuesday, which is documented in each summary, but we can also have extra group meetings, or group meetings with only a few team members on other days of the week. If we plan to have such meetings, we have to write the time and place for them using the text *Next group meeting* followed by the time and place for the full group meeting using the text *Next (full) group meeting*, so that there will be no confusion.

6.2.2 Customer meetings

The customer meetings are not set on a certain day of the week like the group and advisor meetings, so there is no line stating when the next meeting is. When a new meeting is scheduled, the customer contact e-mails the rest of the team informing them about the time and place for the meeting.

6.2.3 Weekly report

The advisor meetings have their own summaries (the template for this is similar to the template for group and customer meetings) but a weekly report is also required. This report will be presented at the advisor meeting, and things we discuss will be part of the new advisor meeting summary.

The weekly report needs to include an agenda for the meeting, meaning what to discuss, and the summaries of the week's meetings, which include group and customer meetings as well as last week's advisor meeting. The team's workload needs to be included, including a description of what each individual has worked on. Other topics include what we have accomplished so far, milestones, what still needs to be done, various problems or issues we might have encountered or are still working on and what our plans are for the next week.

The template in Appendix B has been changed by the advisor, and we no longer have to write who will attend the meetings. If a team member is unable to attend the advisor meeting, he or she has to send the advisor and the rest of the team an e-mail stating the reason for their absence.

6.3 Testing

To make sure that our final product does what it is supposed to do, we need to test it. All the requirements that have been implemented need to be tested to confirm their functionality. Non-functional requirements, or in other words, quality requirements, also need to be tested. These tests need to have a uniform structure and strict guidelines as to how they are being executed to ensure a high level of quality.

Our test plan, and details of each individual test, can be found in chapter 7 and both sprint chapters, where results of tests are stated. The test results will show whether or not the product needs more improvement. We have set limits for the test results and only the best performances will be approved. If the results are under par, we will have to make some adjustments and test again.

Chapter 7

Testing

To be able to properly test our product, we must have a selection of tests we can perform. In this chapter, these tests will be presented and explained. The actual testing will be covered in the sprint chapters (8.3 and 9.4).

This section explains the procedures for the testing [32] of the DoorSign. The purpose of testing is to make sure that the product is working according to the expectations (i.e. functional and non-functional requirements). We also used testing to find bug/errors and fix them. Thus our tests will be based around the functional requirements and the quality requirements.

7.1 Overall test plan

Each sprint will have a testing part. The result of each test will be given in the chapter of each sprint. During the tests, the tester should follow the following points:

- Plan the test
- Implement the test
- Run the test
- Evaluate and document the test result

To document each test, we prepared a template [see table 7.2]. Each test should have an identifier. This identifier is structured as indicates in table 7.1.

Type of test	test identificator
Unit test	UNI-xxx
Modul test	MOD-xxx
System test	SYS-xxx

Table 7.1: Test identificators

7.2 Test planning

This section describes how we are going to organize the tests. We will present the different methods and levels we are going to use and the template that each test should follows.

7.2.1 Test methods

There are three ways of testing: black-box, white-box and grey-box; which is a combination of the two preceding. All testing in this project will be either black-box and white-box testing.

White-box

In white-box testing, the tester has to understand the code and the system. He also needs to have technical knowledge to design and run the tests.

Black-box

In black-box testing, the tester does not need to know, or even understand, the code. The tester only needs to know the possible inputs and the expected outputs associated. The inputs and outputs are based around the requirements.

7.2.2 Testing levels

There are different levels of testing: Low level tests focus on the small parts of the system, whilst high level test are concerned with the system as a whole. Low level tests should be completed before starting on the higher level tests. Therefore, all tests in this chapter will be ordered from low level to high level.

Unit test

Unit test is a low level test. Small units of the project are tested: Classes, methods, variables, etc. to make sure that each small part works as expected.

Modul test

Modul tests are used to test entire modules. Module testing will make sure that the coordination and communication between the different entities works as expected.

System and integration tests

Once each module works as expected, we want to assure that the different modules, that the system consist of, works correctly when put together. This is the purpose of system and integration tests: Testing the coordination and communication between modules and interfaces.

7.2.3 Templates

The table 7.2 presents the template for each test.

Item	**Description**
ID	Identifier of the test [see table 7.1]
Name	Name of the test
Date	Date of the test
Responsible	Name of the person responsible for the testing (by default: the test manager)
Requirements	What requirements will be tested
Precondition	Conditions that have to be met in order to start the test
Execution steps	Steps to perform
Expected result	The expected results of the test
Results	Results of the testing

Table 7.2: Testing template

7.3 Test cases

In this section, we will describe each test. The result of each test will be given in the chapter of each sprint.

Sprint 1:

Item	Description
ID	UNI-001
Name	Display information
Date	During sprint 1
Responsible	Elise Boinnot
Requirements	F1
Precondition	None
Execution steps	1. Launch the application
Expected result	The owner's picture and information (first and last names, title, e-mail, mobile number) are displayed.

Table 7.3: Display information test case

Item	Description
ID	UNI-002
Name	Database
Date	During sprint 1
Responsible	Elise Boinnot
Requirements	F6
Precondition	UNI-001 (Display information)
Execution steps	1. Update *MyName MyLastName MyTitle MyPicture MyMail MyMobileNumber* in the database 2. Launch the application
Expected result	The information displayed reflects the information stored in the database. This means that the application is able to query the database for information.

Table 7.4: Database test case

Item	Description
ID	UNI-003
Name	Display message
Date	During sprint 1
Responsible	Elise Boinnot
Requirements	F1
Precondition	None
Execution steps	1. Launch the application
Expected result	The last message update is displayed. At the first launch of the application, a default message is displayed.

Table 7.5: Display message test case

Item	Description
ID	UNI-004
Name	Update message remotely
Date	During sprint 1
Responsible	Elise Boinnot
Requirements	F2
Precondition	The application is launched
Execution steps	1. Launch Skype Application 2. Send a contact request to the Skype account of the device with the password of that account as the request message 3. Send an instant message
Expected result	The device displays the new message within one minute after it has been sent.

Table 7.6: Update message remotely test case

Sprint 2:

Item	**Description**
ID	MOD-001
Name	Record video and stored it
Date	During sprint 2
Responsible	Elise Boinnot
Requirements	F3 and F4
Precondition	The application is launched
Execution steps	**A** : 1. Press the button *Record a video message* 2. Press the button *Start recording* 3. Talk 4. Stop the recording by pressing the button *Stop* 5. Send the recording by pressing the button *Send* **B** : 1. Press the button *Record a video message* 2. Press the button *Start recording* 3. Stop the recording by pressing the button *Stop* 4. Cancel the recording by pressing the button *Cancel* **C** : 1. Press the button *Record a video message* 2. Press the button *Start recording* 3. Talk
Expected result	**A** : The recording is stored and sent to the owner. **B** : The recording is cancelled (and deleted). Nothing is sent to the owner. **C** : The recording stops after 30 seconds.

Table 7.7: Record video test case

Item	Description
ID	MOD-002
Name	Audio call (alias Video conference)
Date	During sprint 2
Responsible	Elise Boinnot
Requirements	F5
Precondition	The application is launched and the owner is connected to his Skype account
Execution steps	1. Press the button *Call* 2. **a** The owner answers: Talkes fo a while and ends the call **b** The owner rejects the call **c** No response
Expected result	**a** The call was successful (conversation took place) and ended **b** The call was rejected **c** The call was canceled automatically

Table 7.8: Audio call test case

Item	Description
ID	MOD-003
Name	Download messages
Date	During sprint 2
Responsible	Elise Boinnot
Requirements	F7
Precondition	The application is launched.
Execution steps	1. The owner just connects to his skype account
Expected result	The owner can see and play all messages sent to him when he was offline. For each message sent, the date and an hour of the recording is displayed.

Table 7.9: Download messages test case

Item	**Description**
ID	MOD-004
Name	Calendar
Date	During sprint 2
Responsible	Elise Boinnot
Requirements	F8
Precondition	The application is launched.
Execution steps	1. Press the button *Calendar* 2. Press the button *Cancel*
Expected result	After pressing the button *Calendar*, office owners schedule for the current day is displayed. Pressing the *Back* button will return you to the main screen.

Table 7.10: Calendar test case

Chapter 8

Sprint #1

As part of our project, our system has requirements that must be implemented. This requires planning, which is covered in this chapter. When the requirements in our first sprint have been implemented, they need to be tested to make sure they have been fulfilled.

8.1 Planning

Given the small amount of requirements and rather large implementation team there will only be two sprints. In the first sprint, we will to fulfill all high priority requirements and prepare foundation for the second sprint 9. During the second sprint we will finish the implementation part and test the system.

8.1.1 Duration

The start of sprint #1 was planned to October 3th and it lasted for two weeks. The first sprint was started as soon at the software architectural work was done. The first day of the sprint was dedicated to planning and the actual duration of the implementation was 9 days.

8.1.2 Resources

Total estimated resources for this sprint were 180 person hours. Our team has 5 developers and with respect to 25 hours of work during one week, we estimated workload of one programmer to 4 hours per day. The duration of the implementation work was 9 days.

$$5 * 4 * 9 = 180$$

8.1.3 Sprint goal

The goal of the sprint #1 was to fulfill all requirements with high priority. This included the database for storing data about office owners, displaying information on the screen and remote message updating. Quality requirement Q2 was also part of sprint #1. Q2 can be considered as a testing of GUI. Moreover, sprint #1 built the foundation for the next sprint. This means establishing communication with Skype API and preparing the database.

As this is our first sprint, one of our goals is to apply theoretical knowledge about SCRUM methodology in practice and learn how to work as a SCRUM team.

8.1.4 Sprint backlog

The sprint #1 backlog can be seen in table 8.1 together with priorities and estimations of workload. Feature numbers refer to the product backlog in section 4.3.

Feature No.	Description	Priority	Est. hours
1.	Displaying Information	H	25
2.	Update message remotely	H	70
3.	Database	H	35
5.	Usability	H	50

Table 8.1: Sprint #1 backlog

The sprint backlog covers following requirements:

- F1 – The device should be able to display a picture, the first and last names of the owner, the owner's title, and a short message on its screen.
- F2 – It should be possible to update the short messages remotely.
- F6 – The system should have a database with the information about the owner of the office.
- Q2 – You should be able to learn how to use the system by yourself in a short amount of time.

Use cases of these requirements with detailed descriptions can be found in section 4.1.3 – Low-Level Use Cases.

8.2 Implementation

During the first sprint, our goal was to fulfill functional requirements, that we marked with high priority. We broke down these requirement to tasks:

- Display office owners information
- Display rooms information
- Create user friendly interface
- Update office owners message using Skype application
- Create database to store office owners and rooms information

We started off by creating a package and class skelleton resembling our initial architecture and from there we added functionality that was required in this sprint.

8.2.1 Usability and GUI

Java Swing

Swing is a Java GUI framework that provides a predefined set of GUI components, for example buttons, labels, radio buttons, panels and so on. By using this framework, we can easily convey a model-view-controller architecture, by creating multiple views with Swing. Each view represents a possible screen that the visitor must interact with in order to fulfill his/her goal. Figure 8.1 displays how each view is constructed, with their core Swing components.

To represent the functionality, we created a user friendly graphical user interface and all views were added to reflect the future functionality that will be implemented in the next sprint 9. We had the following views implemented during sprint 1:

1. InformationView
2. CallView

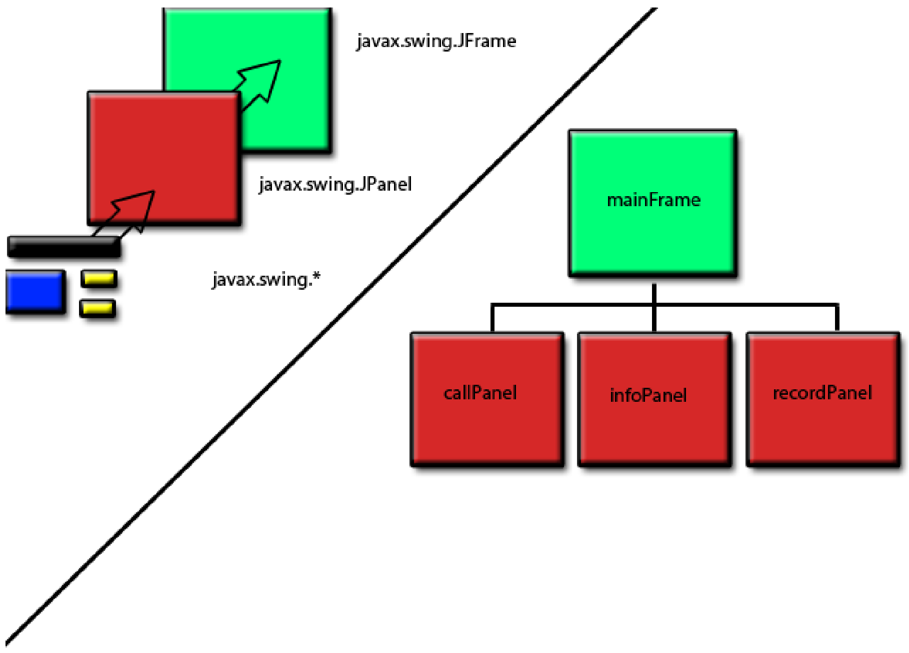


Figure 8.1: GUI – Swing build

3. RecordingView
4. RecordingInformationView

InformationView

This is the main view of the system, that comes up at start-up. All the functionality of our application in this sprint was added to this particular view. Most of the information to fill this view is taken from a database:

- Room number
- Name of the office owner
- Office owner's title
- Office owner's contact information
- Office owner's picture

At the first startup the personal message is set to a default text. The personal message can be updated by sending a Skype IM to the device's Skype account. After the initial change of the message, the default message is changed to the last message sent to the device's Skype account.

CallView

This is intended to facilitate video conference functionality, that will be implmented in next sprint. In this sprint, buttons and actions were added to these views to simulate:

- Call initiation

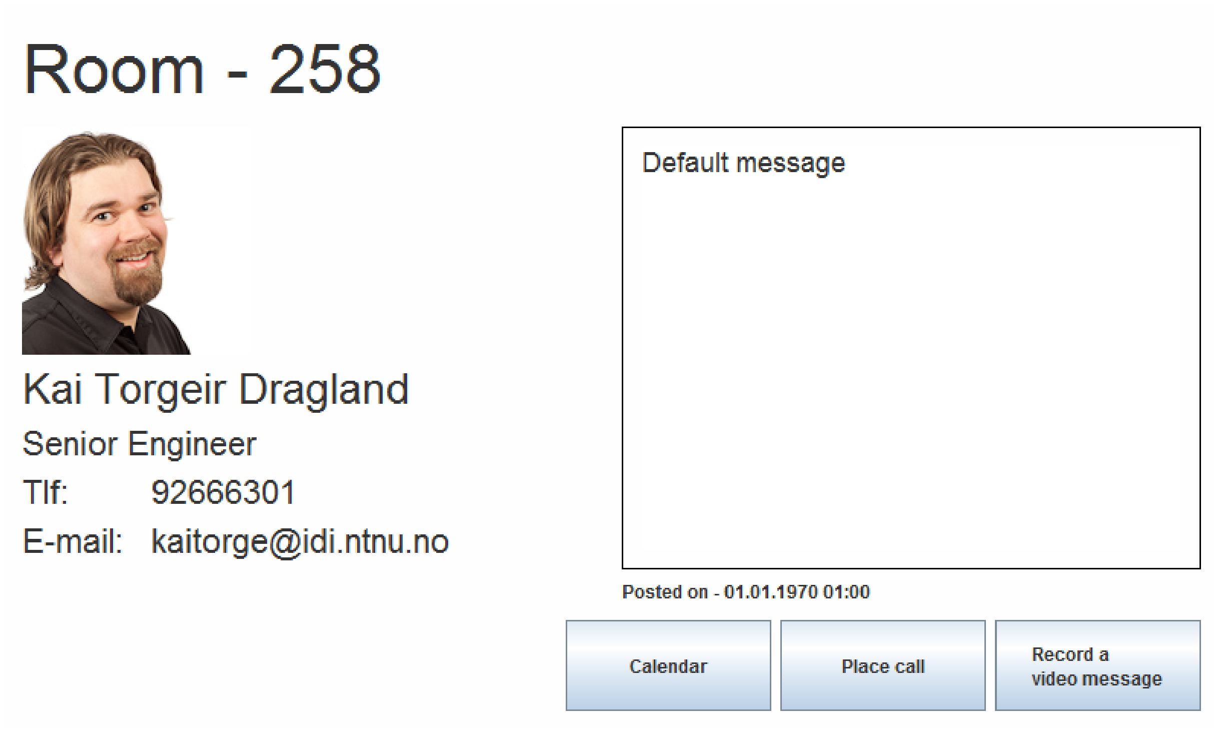


Figure 8.2: The information view

- Call cancelation
- Reception of automated message in case nobody answered the video call

In other words, this view will imitate a phone call, where you try to reach the user, perhaps with a ringing sound.

There are 3 possible outcomes of this function:

- The Owner does not answer the call, e.g. he is busy or indisposed.
- The Owner rejects the call.
- The Owner answers the call.

In the first two points, the visitor gets redirected to the *RecordingInformationView*. If the owner of the office answers the call, the visitor stays in the *CallView* until the call has ended.

RecordingInformationView

This view will display a static message, informing the visitor that the owner is currently unavailable but that the visitor can leave a recorded video message that the owner will receive once he is available.

RecordingView

This view is intended to facilitate video message recording functionality that will be implemented in Sprint #2 9. In this sprint, buttons and actions were added to these views to

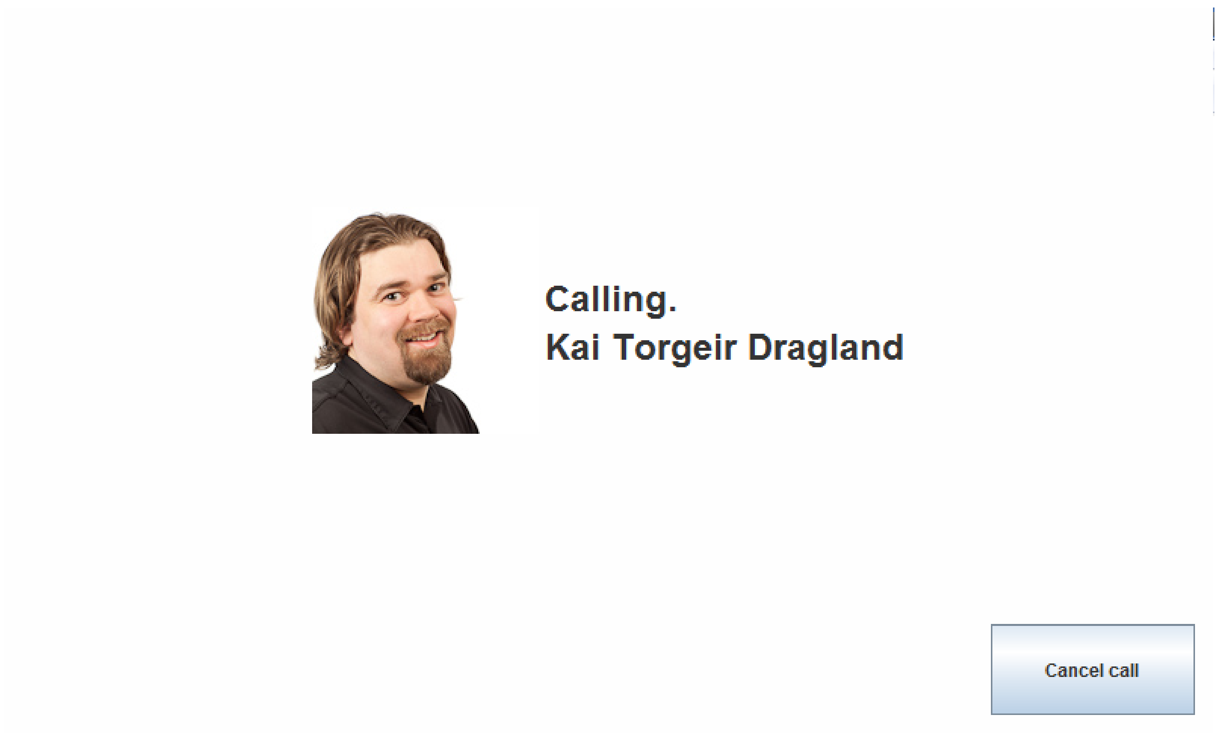


Figure 8.3: The calling view

simulate:

- video message recording and sending to office owner
- video message recording canceling

When actions are cancelled in any of these two views, the application goes back to InformationView.

8.2.2 Database

A database is used to keep various information used by the system. An Entity Relationship diagram is shown in figure 8.4.

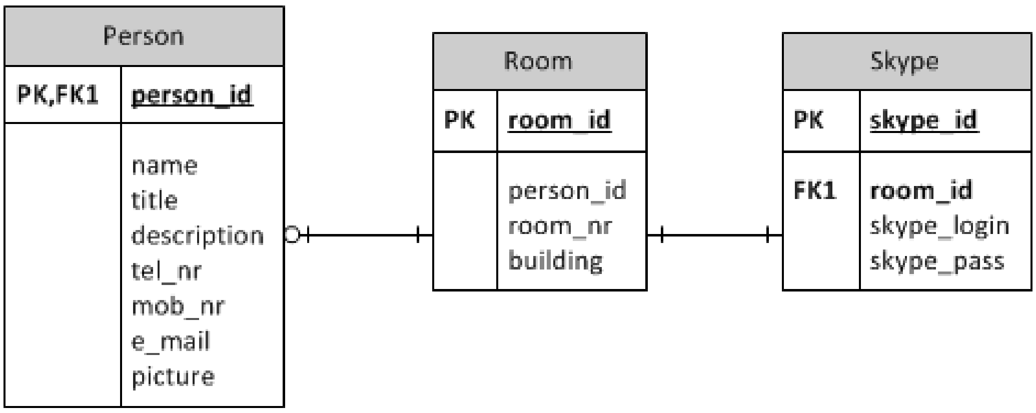


Figure 8.4: Entity-Relationship diagram

Our main database table is *Room*, which contains references to the other two tables. Each room can contain one office owner or can be empty, that is shown by the 0 to 1 connection between Room and Person tables. Each room has only one Skype account, that is represented by the 1 to 1 connection between Room and Skype tables.

Table name	**Field name**	**Field description**
Person	**person_id**	identification number. Used for reference.
	name	name of the office user. This field contains all the names, like first, last and middle names.
	title	title of the office owner. This field contains the scientific degree of the office owner.
	tel_nr	phone number of the office owner. This field contains land-line phone number.
	mob_nr	mobile phone number of the office owner. This field contains personal or work mobile phone number of the office owner.
	e_mail	E-mail of the office owner.
	picture	office owners picture. Link to the picture depicting office owner.
Room	**room_id**	identification number of the table entity. Used for reference.
	person_id	identification nubmer of the user. This fields contains reference to the person table, to identify the person residing in the room.
	room_nr	room number. This field contains actual room number.
	building	building name. This field contains a name of the building.
Skype	**skype_id**	identification number. Used for reference.
	room_id	identification number of the room. This fields contains reference to the room table, to identify skype account of the room.
	skype_login	Skype account login. This field contains login name to the rooms Skype account.
	skype_pass	Skype account password. This field contains password to the rooms Skype account.
	building	building name. This field contains a name of the building.

Table 8.2: Description of ER diagram

Room information is identified by the MAC address of the machine the system is running on. Each room has a corresponding reference to the Person table, that contains personal information about the office owner. Each room has its own Skype account, through which the DoorSign users can communicate to the office owner and vice versa. Detailed description of the ER diagram can be seen from table 8.2.

All information is pulled from the database once, during the start-up of the application. It takes about 16 ms to get the resulting data.

8.3 Testing

In this section we will show the results of the tests done in sprint 1. Descriptions of test cases can be found in chapter 7.3.

8.3.1 Test executed

In sprint #1, we executed the following tests:

- Test case UNI-001
- Test case UNI-002
- Test case UNI-003
- Test case UNI-004

Usability testing has been postponed till after the sprints, when all the functionality has been implemented. '

8.3.2 Test results

The result of the four test cases are shown below.

Item	**Description**
ID	UNI-001 (table 7.3)
Date	13.10.2011
Executioner	Elise Boinnot
Results	Everything works according to the test case: All information displayed correctly.

Table 8.3: Display information test result

Item	**Description**
ID	UNI-002 (table 7.4)
Date	13.10.2011
Executioner	Elise Boinnot
Results	Everything works according to the test case.

Table 8.4: Database test result

Item	Description
ID	UNI-003 (table 7.5)
Date	14.10.2011
Executioner	Elise Boinnot
Results	Everything works according to the test case: A default message is displayed at first launch, otherwise the last message updated is displayed. Informaiton about the message (date, time) is also displayed.

Table 8.5: Display message test result

Item	Description
ID	UNI-004 (table 7.6)
Date	14.10.2011
Executioner	Elise Boinnot
Results	Everything works according to the test case.

Table 8.6: Update message remotely test result

8.3.3 Evaluation of tests

During sprint #1, we executed the test cases UNI-001 (table 7.3), UNI-002 (table 7.4), UNI-003 (table 7.5) and UNI-004 (table 7.6). As seen in table 8.3, table 8.4, table 8.5 and table 8.6, our tests were all successful. So, all implemented functionalities are tested and working.

8.4 Sprint evaluation

8.4.1 Customer feedback

At the end of sprint, we set up a customer meeting to demonstrate our product in its current state and ensure that the application fulfill our customer's expectations. The customer seemed to be very pleased with the results. The only comment was that the font size of the phone number and e-mail address should be bigger, but other than that it looked professional.

8.4.2 Sprint retrospective

During this sprint we started the implementation part of our project using SCRUM methodology. As we had no experience with SCRUM and we expected a lot of problems with SkypeKit, we calculated with the worst case during our estimations. But all implementation work went fine and we managed to implement all features from the sprint backlog. Exception are usability studies which were, after discussion with our advisor, excluded from sprints and will create a separate part of our project. Project plan was appropriately updated.

We encountered several problems, especially in the middle of the sprint we had problems to set up regular meetings because of different timetables. This caused two days delay in integration of modules. Because of overestimations in difficulty of the Skype part, this delay did not cause any problems at the end of the sprint.

One of our goals for this particular sprint was larning to use the SCRUM methodology. This goal was met, but some improvements were presented and will be discussed in the next sections.

During this sprint and especially in the planning part, we managed to divide work based on our skills and compensated for our weaknesses. This affected our effectiveness as a team in a very good manner.

What went well

- No problems during implementation
- Almost all tasks were completed
- Good feedback from our customer
- Good task assignments
- Good planning

Problems encountered

- Problems in finding common time slots for meetings
- Difficulties in cooperation at the beginning of the second part of the sprint
- Not enough detailed sprint backlog limiting reassigning task during the sprint

Improvements suggested

- Set up more SCRUM meetings and focus on collaboration and awareness about the work of other team members
- Do more implementation work within a group work
- Elaborate the sprint backlog

8.4.3 Burn down chart

The burn-down chart of sprint #1 can be seen in figure 8.5. A very sharp drop in the first days of the sprint is caused by, as was already discussed, overestimation of our work (especially Skype part). Our stagnation in the middle of sprint is also clearly visible in the burn-down chart. A package of approximately 15 hours of work left at the end of the sprint is caused by postponing the usability tests till after the end of second sprint 9 because the SCRUM methodology does not allow to change the sprint backlog.

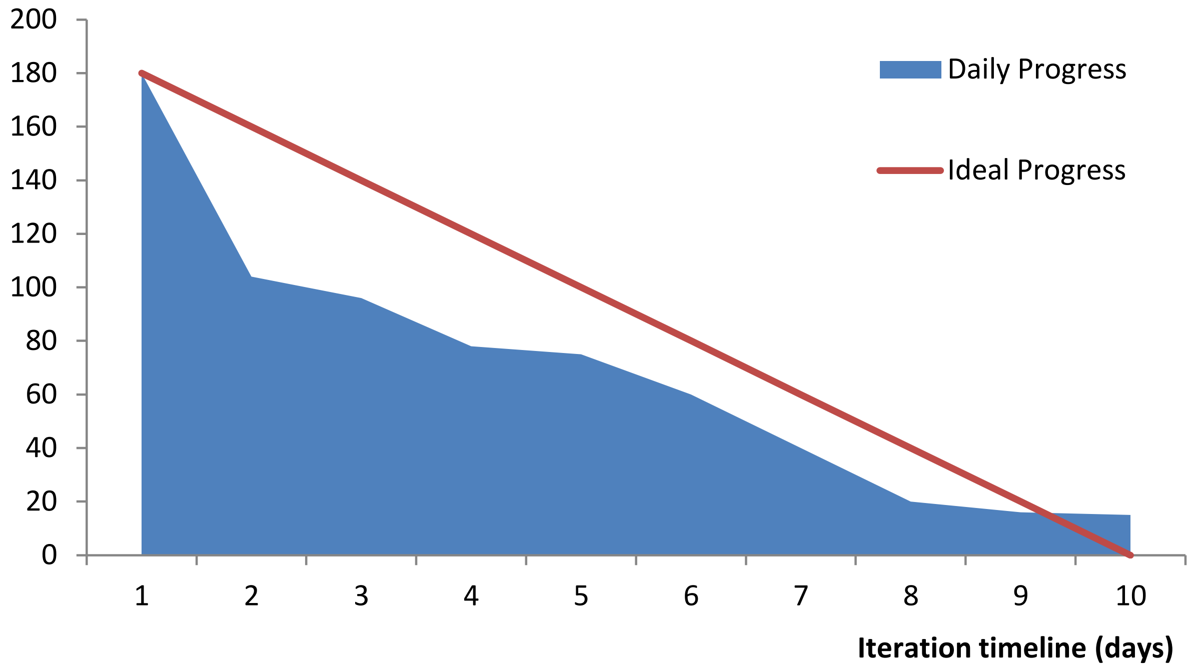


Figure 8.5: Burn down chart for sprint #1

8.4.4 Sprint conclusion

Sprint #1 focused on high-priority requirements and other important components that had to be finished before the second sprint 9 could be started. All the features from the sprint backlog were implemented (with exception of usability study, which was postponed), successfully tested [see section 8.3] and we received a very positive response from our customer. As this was out first sprint, we encountered several problems using SCRUM methodology and we took appropriate measures.

Sprint #1 prepared background for the next sprint by implementing communication between our application and the SkypeKit, which is the basic building block for sprint #2 9.

Chapter 9

Sprint #2

Our final batch of requirements is in our second sprint. This chapter is structured like the previous one, and will include the same sections; planning, implementation and testing.

9.1 Planning

The last sprint focused on implementation of remaining requirements and finishing the application. At the end of sprint #1, we wanted to have a fully tested and documented final product. With the experience from sprint #1, we decided to adjust the way of our estimations and not focus on the worst case, but rather on the most probable case. Moreover, we tried to set up more team meetings and do more of our work within a group. As we have a good feeling from sprint #1, there were no other needs for improvements.

9.1.1 Duration

Sprint #2 was planned to start on October 18th. Its duration was two weeks. This sprint started immediately after finishing sprint #1. The first day of the sprint was dedicated to planning and the actual duration of the implementation was 9 days.

9.1.2 Resources

Total estimated resources for this sprint were the same as for the previous one: 180 person hours. Our team has 5 developers and with respect to 25 hours of work during one week, we estimated workload of one programmer to 4 hours per day. Duration of the implementation work was 9 days.

$$5 * 4 * 9 = 180$$

9.1.3 Sprint goal

The goal of sprint #2 was to finish the implementation work. This includes implementing all remaining requirements and finalizing the application.

9.1.4 Sprint backlog

The sprint #2 backlog can be seen in table 9.1 together with priorities and estimations of our workload. Feature numbers refer to the product backlog 4.3

Feature No.	Description	Priority	Est. hours
6.	Record video	M	25
7.	Store video	H	25
8.	Video conference	M/H	55
9.	Send video messages	L	15
4.	Security Q1: Application restriction	M/H	15
10	Performance	L	15

Table 9.1: Sprint #2 backlog

The sprint backlog covers the following requirements:

- F3 – It should be possible to start recording a short audio and video message. If not stopped manually, the recording will stop automatically after 30 seconds.
- F4 – A recorded message should be stored locally.
- F5 – It should be possible to have a video conference with the owner.
- F7 – When the receiving device (of the owner of the office) has established a connection to the Internet, it should receive all messages sent to him/her during the offline period with additional information, e.g. timestamp.
- Q1 – You should not be able to exit the DoorSign application and access other parts of the device.
- Q3 – You should be able to have a normal conversation through the video conference.

Use cases of these requirements with detailed descriptions can be found in section 4.1.3 – Low-Level Use Cases.

9.2 Implementation

9.2.1 Recording and storing video message

The Mimo device we received from the customer was communicating perfectly with certain software available on the Windows OS. However, official Windows software like Windows Media Player, Sound recording, etc. was not able to detect the hardware [see appendix D.3]. Skype was able to both pick up the video and sound of the device when plugged in into a computer with the Mimo UM-740 drivers obtained from their website [17]. But other components, like Windows Media Player, did not discover the device. The official standard Java Media Framework (JMF) developed by Sun Oracle [20] was only able to detect the the microphone, but not the integrated web-cam.

A fair amount of the hours spent on these two requirements was searching for third-party libraries that would discover both the web-cam and microphone on the device.

The implementation consists of three parts: Recording audio, recording video and merging the audio and video files together. Because there was no third-party software that could detect and record both audio and video we had to record them separately and then merge them together before sending.

The Audio part Seeing that JMF was capable of discovering the microphone, we agreed to use this solution to capture the incoming audio independently of the video.

The Video part The only third-party library that discovered the Mimo web-cam through Java was the LTI-CIVIL (Larson Technologies Inc. Capturing Images and Video In a Library) [15]. CIVIL was only able to capture images, more specifically images of the format *java.awt.image.BufferedImage*. There was no recording option in this library, only receiving X amounts of buffered images per second (X FPS). Therefore we needed a new library capable of storing these buffered images and converting them into a video format.

The Java media library Xuggler [34] allows you to uncompress, modify, and re-compress any media file or stream from Java. This made it possible to convert the series of buffered images into a MPEG-4 Part 14 (MP4) video *on the fly*. After the stream of buffered images ended, the video was stored locally.

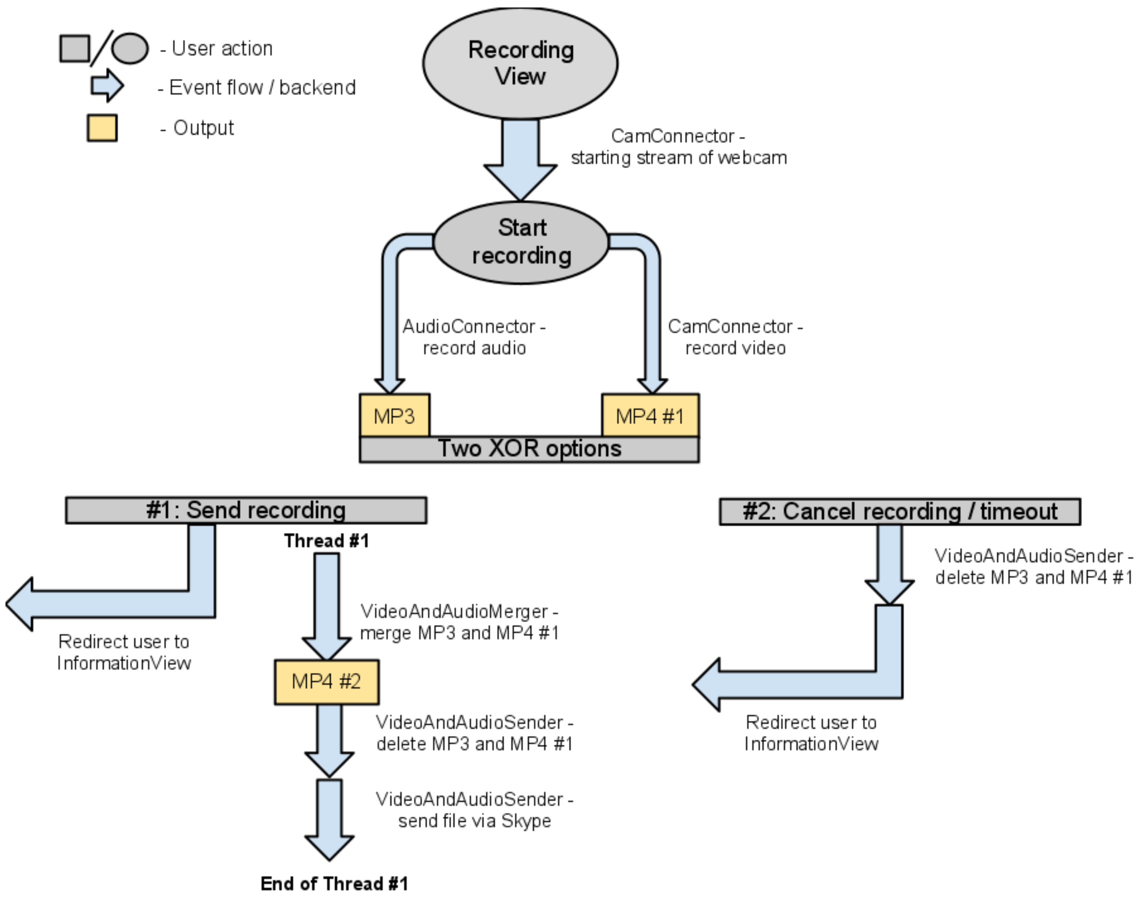

- User action
- Event flow / backend
- Output
Recording View
CamConnector - starting stream of webcam
Start recording
AudioConnector - record audio
CamConnector - record video
MP3
MP4 #1
Two XOR options
#1: Send recording
Thread #1
Redirect user to InformationView
VideoAndAudioMerger - merge MP3 and MP4 #1
MP4 #2
VideoAndAudioSender - delete MP3 and MP4 #1
VideoAndAudioSender - send file via Skype
End of Thread #1
#2: Cancel recording / timeout
VideoAndAudioSender - delete MP3 and MP4 #1
Redirect user to InformationView

Figure 9.1: The work flow of the recording and merging

The execution and merging We were able to concurrently record audio and video through JMF, CIVIL and Xuggler and then store these two files (one MP3 file and one MP4 file) locally on the computer. Then the two files were merged together to create a single file, before we could send the recording over Skype. Xuggler was used for the merging, by braking down the audio and video file to two streams and then creating a new MP4 file with two channels, one for audio and one for video. The merging was successful.

As a result of using this method, the new MP4 file was able to produce both audio and video, but with major synchronization issues. There are also limitations to the size of two files that needs be merged, if they are too large, there is a chance of running out of memory [33].

9.3 Software security

To analize how secure our application is we will be using Microsoft Threat Modeling Process [21]. This is a well documented and popular approach for security analysis and evaluation.

9.3.1 Threat Modeling

Microsoft Threat Modeling Process was chosen to perform threat risk modeling. This is a five step process:

1. Identify Security Objectives
2. Application Overview
3. Decompose Application
4. Identify Threats
5. Identify Vulnerabilities

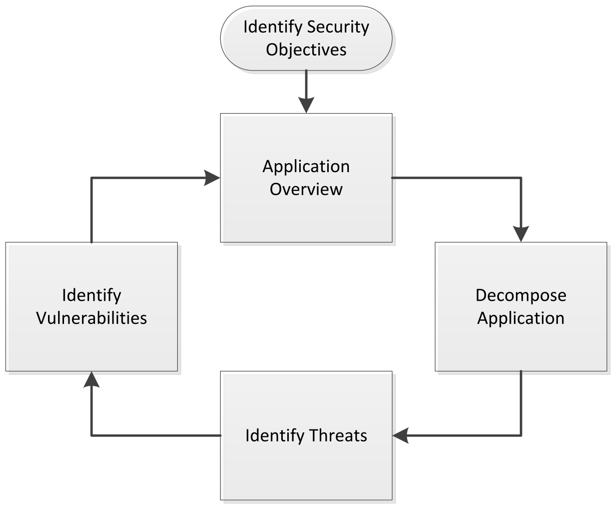


Figure 9.2: Threat Model Flow

Identify security objectives

To ensure that our application is secure, we have to determine what parts of it need to be protected.

- Identity – The office owner's identity does not need to be protected. The private information about the room has to stay private. Since each room has a unique Skype account that communicates with the office owner directly, rooms Skype ID and password need to be protected.
- Reputation – Since somebody already hacked the IDI map monitors, the possibility that people will try to mess with the DoorSign is high. The application has to withstand all known attacks that were used on the IDI map monitors.
- Privacy and regulatory – The application will run on the office owner's personal computer, so in an indirect way the application should protect the office owner's personal data.

Application overview and application decomposition

Our system consists of three parts:

- Database
- Skype Kit
- Application

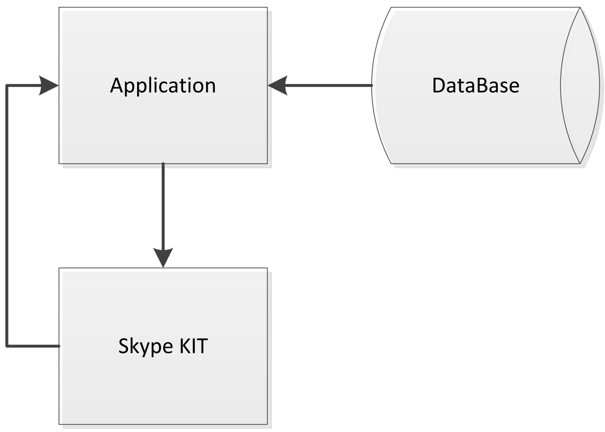


Figure 9.3: Data Flows

The application gets its information from a remote database. The data traveling between the application and the database is not encrypted. The application communicates with SkypeKit [10] by receiving and sending packages to it. This communication provided by SkypeKit's out of our control.

Identifying threats

To identify possible threats we needed to find similar systems to learn of their experience. One of these systems is touch-screen monitors with maps in NTNU IT building. According to the designer of the system, the most obvious way to mess with the system was to simulate right mouse click on the touch-screen monitor using various techniques. Other, more obvious, threats like closing, minimizing, crashing the application were identified, using previous experience of the implementation team.

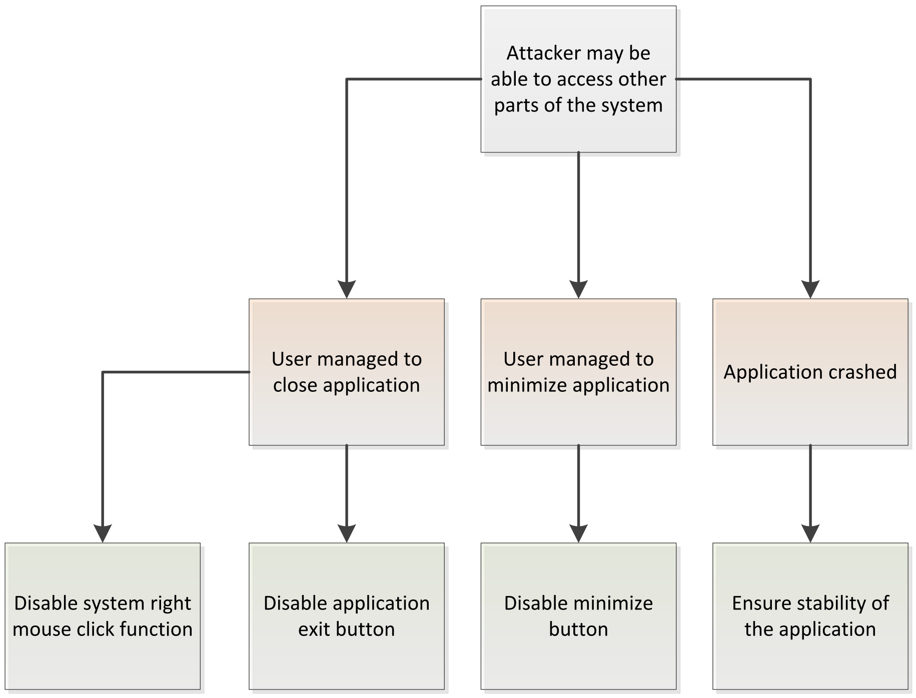

Attacker may be able to access other parts of the system
User managed to close application
User managed to minimize application
Application crashed
Disable system right mouse click function
Disable application exit button
Disable minimize button
Ensure stability of the application

Figure 9.4: Attack on the application

An attacker might be able to access other parts of the system than what is intended, se figure 9.4. Allowing the attacker to access other parts of the system, can expose the office owner's computer and data kept on it.

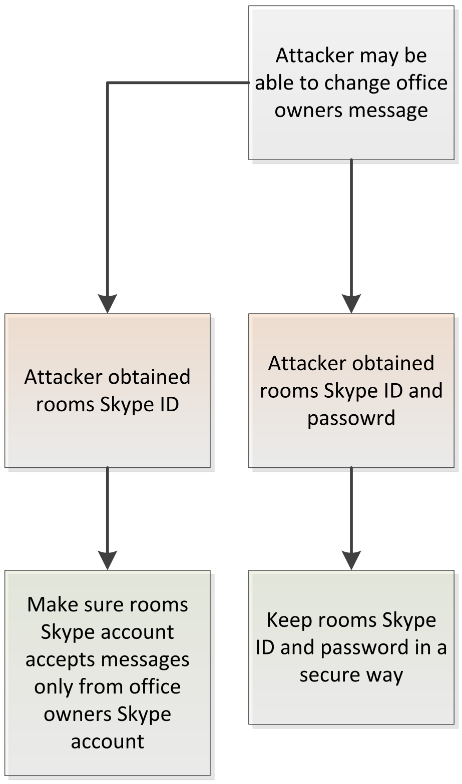

Attacker may be able to change office owners message
Attacker obtained rooms Skype ID
Attacker obtained rooms Skype ID and passowrd
Make sure rooms Skype account accepts messages only from office owners Skype account
Keep rooms Skype ID and password in a secure way

Figure 9.5: Attack on Skype

An attacker might be able to change the office owner's personal message, se figure 9.5. Allowing an attacker to change the office owner's personal message can lead to potential application exposure and loss of reputation.

An attacker might gain access to the database. If the attacker gains access to the database, he will be able to change the office owner's personal information.

Possible attackers (Threat Agents):
It is very important to identify possible attackers to understand how secure the application has to be. The level of security has to be a lot higher if there is a possibility that some attackers will have strong a motivation to attack the system.

Curious Attacker – Security researcher, independent tester or more likely a colleague wanting to brighten up the office owner's day.

Script kiddie – Computer criminals, who attack or deface applications for respect or political reasons.

Motivated attacker – A student who wants to gain access to the lecturers computer to access private information or to take revenge for a bad grade by defacing lecturers DoorSign.

9.3.2 Risk evaluation

Risk evaluation was performed using OWASP Risk Rating Methodology [22]. OWASP was chosen because of good experience with this evaluation method by members of the implementation team. To evaluate the risks we need to look at *threat agent factors* and *vulnerability factors*.

Threat Agent Factors

The goal of this evaluation is to find out how likely the threat agent is to succeed with the attack.

- Skill Level

  The technical skill level of the attacker or attackers.

  Minimal technical skills (1), some technical skills(4), advanced technical skills(7), professional(10).

- Motive

  Level of motivation, based on the reward.

  Small or no reward(1), possible reward(5), high reward(10).

- Opportunity

  What resources are required to exploit this vulnerability of the system.

  Full access to the system or expensive resources are required(1), special access or some resources are required(5), just by interacting with the system(10).

Vulnerability Factors

These are the factors connected to the specific vulnerability. The goal of these estimations is to find out how likely it is for the vulnerability to be found and exploited by the threat agents.

- Ease of discovery

  How easy is it to discover this vulnerability for the selected threat agents?

  impossible(1), difficult(4), easy(7), the vulnerability is obvious(10)

- Ease of exploit

  How easy is it to exploit this vulnerability for the selected threat agents?

  impossible(1), difficult(4), easy(7), the vulnerability exploit is obvious(10)

- Awareness

  How familiar are the threat agents with the vulnerability?

  unfamiliar(1), vulnerability is hidden(4), vulnerability is obvious(7), vulnerability is publicly known(10)

9.3.3 Factors estimating impact

To estimate the impact the attacks may have, we can break down the impact into two categories: *Technical impact* and *business impact*. Even though Business impact ultimately is more important, this project is none profit and it is impossible to evaluate the financial part.

Technical Impact factors

To calculate the impact we need to define impact factors and assign a value from 1-10 to them. The goal is to estimate how much impact the exploited vulnerability will have on the system.

- Loss of confidential information
  How much data can be discovered and how sensitive is it? Some none sensitive data(1), a lot of sensitive data(5), all the data(10)

- Loss of integrity

  How much data can be corrupted and level of corruption?

  Some data to minimal extend(1), a lot of sensitive data to medium extend(5), all the data totally corrupted(10)

- Loss of availability

  How much service would be lost and how dependent on these services in the application?

  Some services are unavailable for short time that don' t disrupt overall application performance(1), most services are disrupted, but the application still runs(5), total disruption of the services, application cannot run without(10)

Determining the severity of the risk

All estimations made, have to consider the worst case scenario. Keeping this in mind, it is important to look at the vulnerabilities from a point of view of experienced, highly motivated attacker.

To calculate the overall likelihood, this formula was used:

Overall likelihood = (Skill Level + Motive + Opportunity + Ease of discovery + Loss of integrity + Awareness)/6

To transform the threat numeric value into HIGH, MEDIUM, LOW table 9.2 is used.

Likelihood and Impact Levels	
1 to < 3	LOW
3 to < 6	MEDIUM
6 to 10	HIGH

Table 9.2: Likelihood and Impact Levels

To determine the overall severity of the risk, we used table 9.3

	Overall risk severity			
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
			Likelihood	

Table 9.3: Severity

An estimation for the likelihood and technical impact has been made for each of the different threats, introduced earlier in this section.

- Attacker may be able to access other parts of the system. Se table 9.4 and 9.5 for the estimation of likelihood and technical impact.

Threat agent factors		
Skill Level	Motive	Opportunity
7	10	10
Vulnerability factors		
Ease of discovery	Ease of exploit	Awareness
10	10	7
Overall likelihood = 9 (HIGH)		

Table 9.4: Access control likelihood

Loss of confidential information	Loss of integrity	Loss of availability
10	6	0

Overall technical impact: 5.33 (MEDIUM)
Overall severity the threat is High.

Table 9.5: Access control technical impact

- Attacker may be able to change office owners message. Se table 9.6 and 9.7 for the estimation of likelihood and technical impact.

Threat agent factors		
Skill Level	Motive	Opportunity
7	6	5
Vulnerability factors		
Ease of discovery	Ease of exploit	Awareness
4	4	4

Overall likelihood = 5 (MEDIUM)

Table 9.6: Data corruption likelihood

Loss of confidential information	Loss of integrity	Loss of availability
3	3	3

Overall technical impact: 3 (LOW)
Overall severity of the threat: Medium

Table 9.7: Data corruption technical impact

- Attacker may gain access to the database. Se table 9.8 and 9.9 for the estimation of likelihood and technical impact.

Threat agent factors		
Skill Level	Motive	Opportunity
7	6	0
Vulnerability factors		
Ease of discovery	Ease of exploit	Awareness
4	4	4

Overall likelihood = 4.8 (MEDIUM)

Table 9.8: Database access control

Loss of confidential information	Loss of integrity	Loss of availability
3	3	7

Overall technical impact: 4.3 (MEDIUM)
Overall severity of the threat: Medium

Table 9.9: Database access control technical impact

9.3.4 Identify and threat vulnerabilities

Using threat evaluation we can now prioritize vulnerabilities and treat them accordingly.

An attacker may be able to access other parts of the system threat can be lessened by disabling monitors functions to simulate right mouse click and by removing applications minimize and close buttons. Since the device has no other means to control the application, this solution should solve this threat.

Attacker may be able to change the office owners message threat has two levels of defense:

Application accepts messages only from the office owners Skype account. Only the office owner knows rooms Skype ID, the ID will be kept on a secure MySQL server.

Attacker may gain access to the database – this threat is harder to treat as the database where the office owners information will be kept is supposed to be secure by default and we have no control over it. A highly skilled attacker might be able to access this database, but it would have impact on other IDI applications and would be treated by IDI system administrators.

9.3.5 Conclusion

Building software security is an ongoing process. Most software that is safe today might not be safe tomorrow, but we feel that our application is safe enough for it is current usage.

9.4 Testing

In this section we will show the result of the tests executed in sprint #2. Descriptions of test cases can be found in chapter 7.3.

9.4.1 Test executed

In sprint #2, we executed the following tests:

- Test case MOD-001
- Test case MOD-002
- Test case MOD-003
- Test case MOD-004

9.4.2 Test results

The result of the four test cases are shown below.

Item	Description
ID	MOD-001 (table 7.7)
Date	08.11.2011
Executioner	Elise Boinnot
Results	Everything works according to the test case

Table 9.10: Record video test result

Item	Description
ID	MOD-002 (table 7.8)
Date	06.11.2011
Executioner	Elise Boinnot
Results	Everything works according to the test case

Table 9.11: Audio call test result

Item	Description
ID	MOD-003 (table 7.9)
Date	08.11.2011
Executioner	Elise Boinnot
Results	Everything works according to the test case

Table 9.12: Download messages test result

Item	Description
ID	MOD-004 (table 7.10)
Date	06.11.2011
Executioner	Elise Boinnot
Results	Everything works according to the test case.

Table 9.13: Calendar test result

9.4.3 Evaluation of tests

During sprint#2, we executed the test cases MOD-001 (table 7.7), MOD-002 (table 7.8), MOD-003 (table 7.9) and MOD-004 (table 7.10). As seen in table 9.10, table 9.11, table 9.12 and table 9.13, our tests were all successful. All implemented functionality has been tested and is working correctly.

9.5 Sprint evaluation

9.5.1 Customer feedback

At the end of the implementation a customer meeting was arranged in order to present the finalized application to our customer. Because of the new requirement [see section 9.5.2] this meeting was especially important. Demonstration revealed several small bugs in the code, but the functionality of the final application was successfully demonstrated. Our customer had several comments regarding button labels and font sizes. The customer also took a usability test (the office owner's perspective). The results were very satisfying. Every task from our questionnaire [see section 10.6.7] was considered as very easy and the final system usability scale score [see section 10.3] was 95 which means grade A.

9.5.2 Requirement change

One of the major issues that influenced this sprint was a requirement change. We were not able to implement the video part of video conference as SkipeKit [10] does not support this functionality yet.

As a result of this discovery, we arranged a meeting with our customer and discussed the problem. We explained why it would not be possible to implement video conference and also announced that this functionality probably will be available in the next release of SkypeKit. Our customer agreed on postponing this requirement to the future work and proposed a new requirement to be implemented instead – calendar functionality. Implementation of calendar functionality was estimated to 40 hours and replaced the video part of our video conference requirement, which was canceled.

9.5.3 Sprint retrospective

Sprint #2 was the last sprint in our project, and all implementation work had to be finished. We focused on the remaining requirements from the product backlog and finalizing the application.

The requirement change which happened during the sprint, influenced our progress. Using the SCRUM methodology, sprint backlog cannot be changed (is frozen) after the sprint planning meeting [29]. Normally, this new requirement would be implemented in the next sprint, but we did not have any more sprints planned. So we decided to add this new functionality to the sprint backlog and implement it during this sprint, despite the possibility of not meeting our schedule.

During the sprint we encountered several issues regarding implementation. The most problematic requirement was recording of a video message. JMF [20], which we had planned to use, did not detect the Mimo monitor's web-cam. After a number of unsuccessful attempt to make it work, we decided to use another library. However, using this new library we had to record video and audio separately, and then merge them together afterwards.

Our implementation team also faced several conflicting project deadlines and other work, which contributed to the delay. Despite the fact that the sprint was not finished according to the plan and had to be prolonged, we implemented all required functionality.

Sprint retrospective meeting revealed some troubles in communication within the implementation team, mainly caused by not attending all meetings. One of the problems was that not everyone had access to the Mimo monitor. This meeting also showed that setting up more regular weekly meetings would be more effective than ad hoc meetings, because team members would consider it as a permanent item in their timetables. This problem would be completely solved if we were able to have daily SCRUM meetings. This would be hard to manage in practice however, because of different timetables and interfering activities of team members.

What went well

- The sprint resulted in the fully working application with all requirements implemented
- Positive customer feedback
- Audio part of video conference and sending video message
- Security requirement
- Good communication with our customer
- Mutual agreement on requirement change with our customer

Problem encountered

- Change of requirements
- Problems with recording message requirement
- A lot of interfering projects and other work
- Communication within implementation team during last several days of the sprint

Improvements suggested

- Have the device available for everyone at the university
- Set up regular meetings instead of ad hoc meetings

9.5.4 Burn down chart

The burn-down chart of sprint #2 can be seen in figure 9.6. The chart reflects our progress with the implementation work. The two days delay was caused by the requirement change and by interfering activities of team members.

9.5.5 Sprint conclusion

During sprint #2 all requirements, except the cancelled one, were implemented from the sprint backlog, despite several days delay. Risk of delay during implementation work caused by interfering activities of team members was recognized during risk analysis. As a measure, project plan contained appropriate buffer and therefore this delay did not cause any problems for the project as a whole.

Our customer had several small objections regarding the visual part of the application, but these were minor problems and were adjusted. Otherwise the feedback was positive.

Sprint resulted in the fully working application with all the requirements implemented, except the cancelled one, which was now ready to be used in the the usability testing phase.

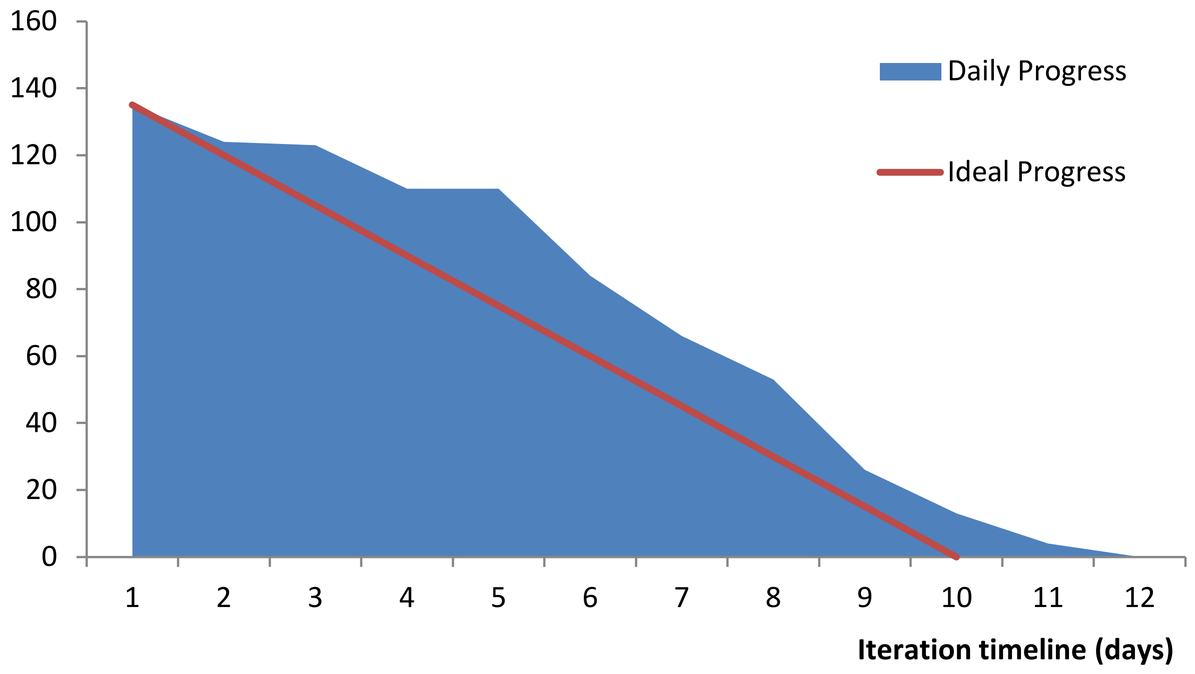


Figure 9.6: Burn down chart for sprint #2

Chapter 10

Usability study

To determine whether or not the product is a success, we need to determine if users can use this product to it's full extent, creating the product is only one part of the job. The another crucial part is to make sure it can actually be used by the end-user. The product should be easy to use, easy to learn and, most importantly, it should be useful. Based on ISO's definitions of usability [30], this means that users should not have any difficulties figuring out how to use the product and once they have learned how, they should be able to perform tasks quickly.

To make sure this is the case with our product, we have developed usability tests for our two groups of users. Most of our test subjects will execute the usability test from the visitor's perspective, while the rest will execute the usability test from the office owner's perspective. During the testing, members of our team will act as the office owner and visitor, respectively. By executing these tests and filling in the questionnaires, our test subjects will be a tremendous help in verifying that our product meets our standards.

At the beginning of this chapter we will describe our goals of the usability study itself and how we approached the process of developing tests that matched our needs. We will also introduce a standard way of calculating and interpreting the questionnaire scores, with an explanation of which scores we think of as satisfactory.

Our usability test plans will start with the plan from the visitor's point of view. In this section we will explain which groups of visitors we imagine will use our product. The practical part of our usability study follows this section. Here we will describe what kind of materials we will need for the testing and the procedure itself, i.e. how the test subjects will be prepared before the testing starts and details of how the testing will be carried out. After this follows a list of tasks that the test subject will try to finish during the testing. Our usability metrics section shows how we will interpret the test subjects' answers to the questionnaires. Our success criteria are linked to these questionnaires and the tasks given to the test subjects.

The next section contains the test plan from the office owner's point of view with details about materials needed and the procedure used. Then follows a description of the tasks that the test subjects who are playing the role of the office owner will be given. Our roles during these tests, along with the usability metrics and success criteria, will be described, summing up the chapter. This section also includes a questionnaire specifically designed for the office owner that the test subjects will fill out.

10.1 Goals

The desired result of our implementation is a working prototype. We will not be executing another sprint modifying the application based on the results of this usability test, therefore, the main goal of our usability testing is to assess the product and find out whether or not it is valuable for both visitors and the office owner. The execution of the usability scenario and its evaluation will show us the limitations of the product and reveal potential future work and development.

10.2 Approach

The usability of the DoorSign can be observed from two different perspectives:

- The visitors perspective
- The office owners perspective

The most important factor to test during our usability study, from our visitor's perspective, is the immediate understanding of the functionality and features the application and to be able to use it without any assistance.

For the owner of the office it is important to have an integrated, easy-to-use solution that enables communication with visitors.

The usability study is divided into two parts, each reflecting one of the perspectives. The first part focuses on visitors and the second one to the office owner.

10.3 System Usability Scale (SUS)

As a part of our usability study we used the System Usability Scale, this provides a robust and reliable evaluation tool, which is considered as an industry standard [24]. System usability scale is a simple ten-item scale, giving a global assessment of the systems usability [6]. The SUS is used after the test participant has had the opportunity to use the system, but before any discussion took place. The result of SUS usage is a score ranging from 0 to 100. Interpretation of this score can be seen in figure 10.1.

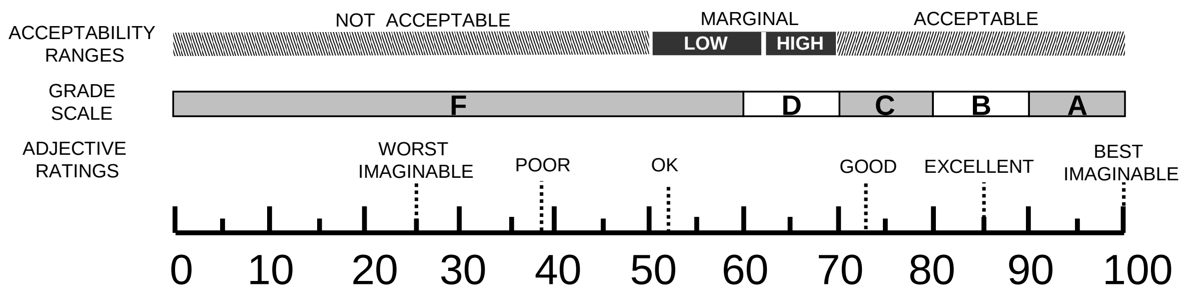


Figure 10.1: SUS score [3]

10.4 List of terms

This section describes terms used in the usability study.

- The (DoorSign) application – the software application developed during this project
- The device – the Mimo monitor (Mimo UM-740)
- The product – the Mimo monitor together with the DoorSign application
- The (DoorSign) solution – the Mimo monitor together with the DoorSign application
- The office owner – the occupant of an office associated with a device asd

10.5 Usability test plan designed for visitors

This section describes a test plan (scenario) for conducting a usability study. The basic outline of the plan uses the *Usability Test Plan* template [27]. The test itself consists of three major parts:

- Simulation of real-life usage of the system
- Assessment of easy-of-use of particular tasks (reflect requirements)
- SUS assessment

10.5.1 Participants

The expected end-users of our DoorSign solution (from the visitors perspective) are students, university staff and external visitors. The largest expected group are students coming to visit a professor.

We cannot expect that visiting students and professors will be from the ICT domain (IDI department, respectively). Therefore, the subjects of the usability testing will be mostly students and some professors from NTNU in any field of study, who are not connected to the application in any way.

There will be approximately 15 test subjects in our usability study and they will participate individually. Each team member will be responsible for two subjects, where at least one will be from a different field of study than ICT. As there are three international students in our team, we will also have several foreign students participating in the usability test.

The participants do not require any prior knowledge of our product or other software systems to participate in the usability test.

10.5.2 Methodology

The methodology we are utilizing will be explained through the training prior to the test, then the location, the equipment, the procedure and finally the tasks.

Training

There will be no training provided, either for the user or the test personell. The test is based on zero knowledge of the application (participants may have some information about the device).

Location

There will be more than one location for the testing of the device, depending on the group of participants. Most of the testing, if not all, will be done on the Gloshaugen campus. The location of the testing will be a random place, capable of supporting the required equipment.

Equipment

In order to be able to run the usability test, the following equipment must be available:

- Personal computer with all the necessary drivers and the Skype application installed.
- Personal computer operator, who will simulate the office owner's activities through the Skype application.
- The DoorSign device (Mimo UM-740).
- The DoorSign application with all interaction functionality implemented (can be in a form of prototype).
- Internet connection.
- Questionnaire in a paper form.

Procedure

1. Participants are welcomed by the test facilitator.
2. Participants are given a basic overview of the test. This includes information about usability testing in general and about the purpose of this particular test. The participants are given a brief explanation of the test procedure and are informed of the duration of the test. The term *DoorSign* will not be used in the briefing. The participants will not be given any information revealing the purpose of the application itself.
3. Participants are given the device with the application installed without any additional information.
4. Participants are told that they have time to explore the application.
5. Participants are given prepared forms (which includes the questionnaire). This step must not be carried out before step 4 as the questionnaire might be able to reveal some aspects of the solution.
6. Participants are asked to write what the purpose of the solution is in the prepared form.
7. Participants are asked to write down the discovered functionalities of the solution in the prepared form.

8. Participants are told to perform a particular task [see table 10.1] and fill in the corresponding part in the form. Their behavior while performing this action is observed and measured.
9. The previous step is repeated until all tasks are carried out.
10. Participants are asked to fill in the rest of the questionnaire.
11. Facilitator expresses the team's gratitude towards the participants for time spent on testing.

Tasks

Table 10.1 gives an overview of the actions that will be carried out during the usability test. All tasks have the prerequisite that the application is fully functional.

Task ID	Description
1	Identify the office owner and his contact information
2	Read the office owner's status message
3	Make a call to the office owner
4	Make a call to the office owner and cancel the call after two (2) seconds of dialog
5	Leave a video message to the office owner
6	Start recording a video message for the office owner and cancel the recording after two (2) seconds
7	Find out if the office owner has/had a free time slot at 1 PM today

Table 10.1: Task list

Template

The description of each task will follow the template given in table 10.2.

Item	Description
Task ID	Id of the task
Name	Name of the task
Preconditions	Conditions that have to be met in order to start the test
Execution steps	Steps to perform
Expected result	The expected results of the test

Table 10.2: Task description template

Description of the tasks

Item	Description
Task ID	1
Name	Identify the office owner and his contact information
Preconditions	Database filled with testing data
Execution steps	1. Participants are not allowed to touch the device
Expected result	Participant identified the office owner and his contact information

Table 10.3: Task 1 description

Item	Description
Task ID	2
Name	Read the office owner's status message
Preconditions	Status message was updated
Execution steps	1. Participants are not allowed to touch the device
Expected result	Participant read the status message of the office owner

Table 10.4: Task 2 description

Item	Description
Task ID	3
Name	Make a call to the office owner
Preconditions	Person representing the office owner is ready to answer the call
Execution steps	1. Press *Call* button 2. Wait for the office owner to pick up
Expected result	Participant made a successful call for the office owner

Table 10.5: Task 3 description

Item	Description
Task ID	4
Name	Make a call to the office owner and cancel the call after two seconds of dialog
Preconditions	Person representing the office owner is ready to answer the call
Execution steps	1. Press *Call* button 2. Wait for the office owner to hang up 3. Wait for 2 seconds 4. Press *Cancel* button
Expected result	Participant started dialoging with the person representing the office owner and canceled the call after 2 seconds

Table 10.6: Task 4 description

Item	Description
Task ID	5
Name	Leave a video message to the office owner
Preconditions	none
Execution steps	1. Press the button *Record a video message* 2. Press the button *Start recording* 3. Speak loudly while looking into the camera and wait for the recording to stop
Expected result	Participant was able to leave a video message to the office owner

Table 10.7: Task 5 description

Item	Description
Task ID	6
Name	Start recording a video message for the office owner and cancel the recording after two seconds
Preconditions	none
Execution steps	1. Press the button *Record a video message* 2. Press the button *Start recording* 3. Speak loudly while looking into the camera for 2 seconds 4. Press the *Cancel* button
Expected result	Participant was able to start recording a video message for the office owner and canceled the recording after two seconds

Table 10.8: Task 6 description

Item	Description
Task ID	7
Name	Find out whether the office owner has / had a free time slot at 1 pm today
Preconditions	none
Execution steps	1. Press the button *Calendar* 2. Observe the today's timetable for the office owner 3. Press the *Back* button
Expected result	Participant was able to see if the office owner has/had a free time slot at 1 PM today

Table 10.9: Task 7 description

10.5.3 Estimated time of the test

Estimated time for a participant to complete a usability test is 8 minutes.

10.5.4 Roles

An individual may play multiple roles, and tests may not require all roles to be present. The list of roles follows:

- Facilitator – Conducts the test and provides guidance to the test subject and responds to the test subjects' questions or requests.
- Data logger – Records the test subject's actions and comments and takes notes.
- Office owner representative – Answers a test subject's calls, updates messages.
- Observers – Assist the data logger in taking notes. They are not allowed to speak during testing.
- Test subjects – Carry out a set of required actions to test the usability of the product.

10.5.5 Usability Metrics

Usability metrics allows us to measure user performance against specific performance goals, i.e. the actual result from the usability test versus our initial expectations . A list of usability metrics with a description follow:

1. Successful recognition of the application's purpose without any knowledge of it, described in a percentage.
2. Successfully recognized functions of the application without any prior knowledge of them, described in a percentage.
3. Users' rating of particular tasks regarding difficulty of the operation
4. SUS evaluation

Percentage of successful recognition of the application's purpose and final location without any knowledge of it
As described in the methodology section [10.5.2], test subjects are given the product without any additional information about it's purpose. After 30 seconds of examining it, they will briefly describe the purpose of the solution. Their descriptions are evaluated against the following statement:

The purpose of this solution is to serve as an interactive <u>door sign</u>.

The evaluation of the test subject's answers will be done within the whole team. The entire team must unanimously agree on whether or not the participant fully understood the purpose of the solution.

The metric expresses the number of correctly identified functions of the solution.

Percentage of successfully recognized functions of the application without any prior knowledge of them
Participants write down functionalities of the solution. Their responses are evaluated against the following list:

- Identify the office owner and his contact information

- Read the status message left by the office owner
- Gain information about the office owner's time table from the calendar
- Make a call to the office owner
- Leave a video message

The evaluation of the test subjects' answers will be done within the whole team. The entire team must unanimously agree on whether or not the participant correctly identified the individual functions of the application.

The metric expresses percentage of successfully recognized roles of the solution.

Users' rating of particular tasks regarding difficulty of the operation
Participants are given tasks to perform and they will assess the difficulty of the operation. A five-point rating scale with appropriate labels will be used [see questionnaire in section 10.5.7].

The metric expresses the difficulty of each task. Tasks are evaluated independently.

SUS evaluation
Participants are asked to answer ten question from System Usability Scale [see section 10.5.7]. SUS calculation procedure [6]:

1. Sum the score contributions from each item (range from 0 to 4)
    - To For items 1,3,5,7,and 9 the score contribution is the scale position minus 1
    - For items 2,4,6,8 and 10, the contribution is 5 minus the scale position
2. Multiply the sum of the scores by 2,5 to obtain the overall value

10.5.6 Success criteria

Goals defining whether or not usability was achieved is evaluated based on the output from the usability study/testing. The numbers in the following descriptions refer to the usability metrics in section 10.5.5.

1. The percentage of successful recognition of the application's purpose should be more than 80%.
2. The percentage of successful recognition of the functions of the application should be more than 80%.
3. More than 80% of the answers regarding the tasks should be in the *very easy* or *easy* part of the questionnaire.
4. SUS score should be higher than 70 [see figure 10.1 for meaning].

10.5.7 Questionnaire

The questionnaire consists of three parts. The first one is separate, because questions and tasks from the second part would influence the answers. The first part's purpose is to provide a place for responds to the first section of the usability test where participant has no knowledge about the application [details in section 10.5.2].

The second part utilizes a five-point rating scale with appropriate labels and will be used for evaluating tasks [see table 10.1]. The participants will circle the appropriate number on the scale, as well as write any comments they may have in the *comments* section.

The third part utilizes System Usability Scale [see section 10.3].

Questionnaire part 1

What is the intended purpose of the application?

. . . .

. . . .

List all functionalities of the device.

. . . .

. . . .

. . . .

. . . .

Questionnaire part 2

Tasks / question:	Very difficult ⟵				Very easy ⟶
1. Identify the office owner and his contact information.	☐	☐	☐	☐	☐
2. Read office owner's status message.	☐	☐	☐	☐	☐
3. Make a call to the office owner.	☐	☐	☐	☐	☐
4. Make a call to the office owner and cancel the call after two seconds of dialing.	☐	☐	☐	☐	☐
5. Leave a video message for the office owner.	☐	☐	☐	☐	☐
6. Start a recoding of a message for the office owner and cancel the recording after two seconds.	☐	☐	☐	☐	☐
7. Find out whether the office owner has / had a free time slot at 1 pm today.	☐	☐	☐	☐	☐
	Strongly disagree ⟵				Strongly agree ⟶
The system would be useful for me as a student.	☐	☐	☐	☐	☐
Using the system would increase the efficiency of my daily work.	☐	☐	☐	☐	☐
The idea of an interactive door sign is good.	☐	☐	☐	☐	☐

Comments:

. . . .
. . . .
. . . .
. . . .
. . . .

Table 10.10: Task list questionnaire – visitors' perspective

Questionnaire part 3

Tasks / question:	Strongly disagree ⟵				Strongly agree ⟶
I think that I would like to use this system frequently.	☐	☐	☐	☐	☐
I found the system unnecessarily complex	☐	☐	☐	☐	☐
I thought the system was easy to use.	☐	☐	☐	☐	☐
I think that I would need the support of a technical person to be able to use this system	☐	☐	☐	☐	☐
I found the various functions in this system were well integrated.	☐	☐	☐	☐	☐
I thought there was too much inconsistency in this system.	☐	☐	☐	☐	☐
I would imagine that most people would learn to use this system very quickly.	☐	☐	☐	☐	☐
I found the system very cumbersome to use.	☐	☐	☐	☐	☐
I felt very confident using the system	☐	☐	☐	☐	☐
I needed to learn a lot of things before I could get going with this system	☐	☐	☐	☐	☐

Comments:
. . . .
. . . .
. . . .
. . . .
. . . .

Table 10.11: SUS questionnaire – visitors' perspective

10.6 Usability test plan designed for office owners

This section describes a test plan for conducting a usability study for the office owner perspectives. The basic outline of the plan uses the *Usability Test Plan* template [27].

10.6.1 Participants

The expected end-users of our DoorSign solution (from the point of view of the office owner) are university staff, especially from IDI department.

We do not require any prior knowledge of our product or other software systems for participating in the usability test.

10.6.2 Methodology

This section describes the methodology we will use for testing usability. For describing this methodology, we will first explain the training prior to the test, then the location, the equipment, the procedure and finally the tasks.

Training

Training consists of reading the user manual of our application.

Location

There will be more than one location for the testing of the device, depending on the group of participants. Most of the testing, if not all, will be done on the Gløshaugen campus. The location of the testing will be a random place, capable of supporting the required equipment.

Equipment

In order to be able to run the usability test, the following equipment must be available:

- Personal computer with the Skype application installed
- Personal computer connected to the *DoorSign* installed, with all the necessary drivers.
- The DoorSign device (Mimo UM-740).
- The DoorSign application with all interaction functionality implemented (can be in a form of prototype).
- Internet connection.
- Questionnaire in a paper form.
- User guide for the office owner.

Procedure

1. Participants are welcomed by the test facilitator
2. Participants are given a basic overview of the test. This includes information about usability testing in general and about the purpose of this particular test. The participants are given a brief explanation of the test procedure and are informed of the duration of the test. The participants will be given some information revealing the purpose of the application itself.
3. Participants are given the office owner user guide and asked to read it.
4. Participants are given the personal computer with Skype application installed.

5. Participants are given prepared forms (which includes the questionnaire).
6. Participants are told to perform a particular task [see table 10.12] and fill in the corresponding part the form. Their behavior while performing this action is observed and measured.
7. The previous step is repeated until all tasks are carried out.
8. Participants are asked to fill in the rest of the questionnaire.
9. Facilitator expresses the team's gratitude towards the participants for time spent on testing.
10. The test is finished.

If the participants do not have a Skype account and/or a Google account a default one is given to them.

Tasks

Table 10.12 gives an overview of the actions that will be carried out during the usability test. All tasks have the prerequisite that the application is fully functional.

Task ID	Description
1	Add *DoorSign* contact (name: tdt4290, password: tdt4290)
2	Update the personal message
3	Answer a call
4	Receive a video message
5	Remove door sign contact

Table 10.12: Task list

Template

The description of each task will follow the template given in table 10.2.

Description of the tasks

Item	Description
Task ID	1
Name	Add door sign contact
Preconditions	None
Execution steps	1. Press *Add contact* in Skype 2. Find username *tdt4290* 3. Type *tdt4290* (the password of the *DoorSign's* Skype account) as the request message 4. Press *Send request*
Expected result	Participant added the door sign account in the contact list

Table 10.13: Task 1 description

Item	Description
Task ID	2
Name	Update a message
Preconditions	None
Execution steps	1. Open a conversation with the *DoorSign's* Skype account 2. Enter a message in the text field 3. Send the message
Expected result	Participant was able to send a message and receive a reply from the *DoorSign's* that it has been updated.

Table 10.14: Task 2 description

Item	Description
Task ID	3
Name	Answer a call
Preconditions	Person representing the visitor calls the owner of the office
Execution steps	1. Click on *Answer call*
Expected result	Participant was able to answer the call as he would any other Skype call.

Table 10.15: Task 3 description

Item	Description
Task ID	4
Name	Receive a video message
Preconditions	Person representing the visitor left a video message
Execution steps	1. Accept the video file
Expected result	Participant was able to receive the video message

Table 10.16: Task 4 description

Item	Description
Task ID	5
Name	Remove door sign contact
Preconditions	None
Execution steps	1. Open a conversation with the *DoorSign's* Skype account 2. Send *tdt4290* (*DoorSign's* Skype account password) as an *Instant Message*
Expected result	Participant was able to remove the *DoorSign* Skype account from his contact.

Table 10.17: Task 5 description

10.6.3 Estimated time of the test

Estimated time for a participant to complete a usability test is 8 minutes.

10.6.4 Roles

In this section we will list all the roles in our usability test plan. An individual may play multiple roles and tests may not require all roles to be present. The list of roles follows:

- Facilitator – Conducts the test. Provides guidance to the test subject and responds to the test subjects' questions or requests.
- Data logger – Records the test subject's actions and comments and takes notes.
- Visitor representative – Call the office owner, record video messages and send them to the office owner
- Observers – Assist the data logger in taking notes. They are not allowed to speak during testing.
- Test subjects – Carry out a set of required actions to test the usability of the product.

10.6.5 Usability Metrics

Usability metrics allow us to measure user performance against specific performance goals. The list of usability metrics and their description follow:

1. Users' rating of particular tasks regarding difficulty of the operation
2. SUS evaluation

Users' rating of particular tasks regarding difficulty of the operation
Participants are given tasks to perform and they will assess the difficulty of the operation. A five-point rating scale with appropriate labels will be used [see questionnaire in section 10.6.7].

The metric expresses the difficulty of each task. Tasks are evaluated independently.

SUS evaluation
Participants are asked to answer ten question from System Usability Scale [see section 10.5.7]. SUS calculation procedure [6]:

1. Sum the score contributions from each item (range from 0 to 4)
   - To For items 1,3,5,7,and 9 the score contribution is the scale position minus 1
   - For items 2,4,6,8 and 10, the contribution is 5 minus the scale position
2. Multiply the sum of the scores by 2,5 to obtain the overall value

10.6.6 Success criteria

Goals defining whether or not usability was achieved were prepared based on metrics. The numbers in the following descriptions refer to the usability metrics in section 10.6.5.

1. More than 80% of the answers regarding the tasks should be in the *very easy* or *easy* part of the questionnaire.
2. The ratio between the number of successful operations compared to total operations should not be below 8/10.
3. SUS score should be higher than 70 [see figure 10.1 for meaning].

10.6.7 Questionnaire

The questionnaire consists of two parts. The first part utilizes a five-point rating scale with appropriate labels and will be used for evaluating tasks [see table 10.12]. The participants will circle the appropriate number on the scale, as well as write any comments they may have in the *comments* section.

The second utilizes System Usability Scale [see section 10.3].

Questionnaire part 1

Tasks / question:	Very difficult $\longleftarrow$				Very easy $\longrightarrow$
1. Add door sign contact (name: tdt4290, password: tdt4290).	☐	☐	☐	☐	☐
2. Update a message.	☐	☐	☐	☐	☐
3. Answer a call.	☐	☐	☐	☐	☐
4. Receive a video message.	☐	☐	☐	☐	☐
6. Remove door sign contact.	☐	☐	☐	☐	☐

Comments:

. . . .
. . . .
. . . .
. . . .
. . . .

Table 10.18: Task list questionnaire – office owner's perspective

Questionnaire part 2

Tasks / question:	Strongly disagree ⟵				Strongly agree ⟶
I think that I would like to use this system frequently.	☐	☐	☐	☐	☐
I found the system unnecessarily complex	☐	☐	☐	☐	☐
I thought the system was easy to use.	☐	☐	☐	☐	☐
I think that I would need the support of a technical person to be able to use this system	☐	☐	☐	☐	☐
I found the various functions in this system were well integrated.	☐	☐	☐	☐	☐
I thought there was too much inconsistency in this system.	☐	☐	☐	☐	☐
I would imagine that most people would learn to use this system very quickly.	☐	☐	☐	☐	☐
I found the system very cumbersome to use.	☐	☐	☐	☐	☐
I felt very confident using the system	☐	☐	☐	☐	☐
I needed to learn a lot of things before I could get going with this system	☐	☐	☐	☐	☐

Comments:

. . . .

. . . .

. . . .

. . . .

. . . .

Table 10.19: SUS questionnaire – visitor's perspective

10.7 Usability study results and evaluation

10.7.1 Visitor's perspective

For our usability study, 14 people were asked for the visitor's perspective. As described in the test plan designed for visitors (section 10.5), the questionnaire were divided in three parts. In this section, we will describe for each part the result of the test.

Part 1 – Purpose and functionalities

To test if the end-user understood the purpose of the application, we asked two questions. The first one required the participants to describe the purpose of the application after exploring it. The second one required them to list the functionality of the application.

As described in section 10.5.5, all answers to the first questions were discussed by the team to conclude whether or not the participant understood the purpose of the system. The results for this question are presented in figure 10.2.

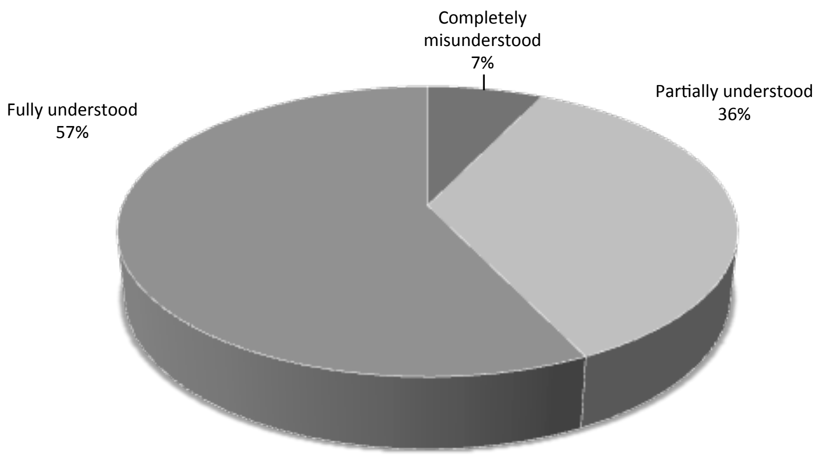


Figure 10.2: End-user's comprehension of the application's purpose (Part 1.1)

As we can see in figure 10.2, only 57% of the participants successfully recognized the purpose of the application. This is less than expected in section 10.5.6 (more than 80%).

For the second question, figure 10.3 presents the results. We can see how often each functionality was recognized correctly, wrongly or was missing.

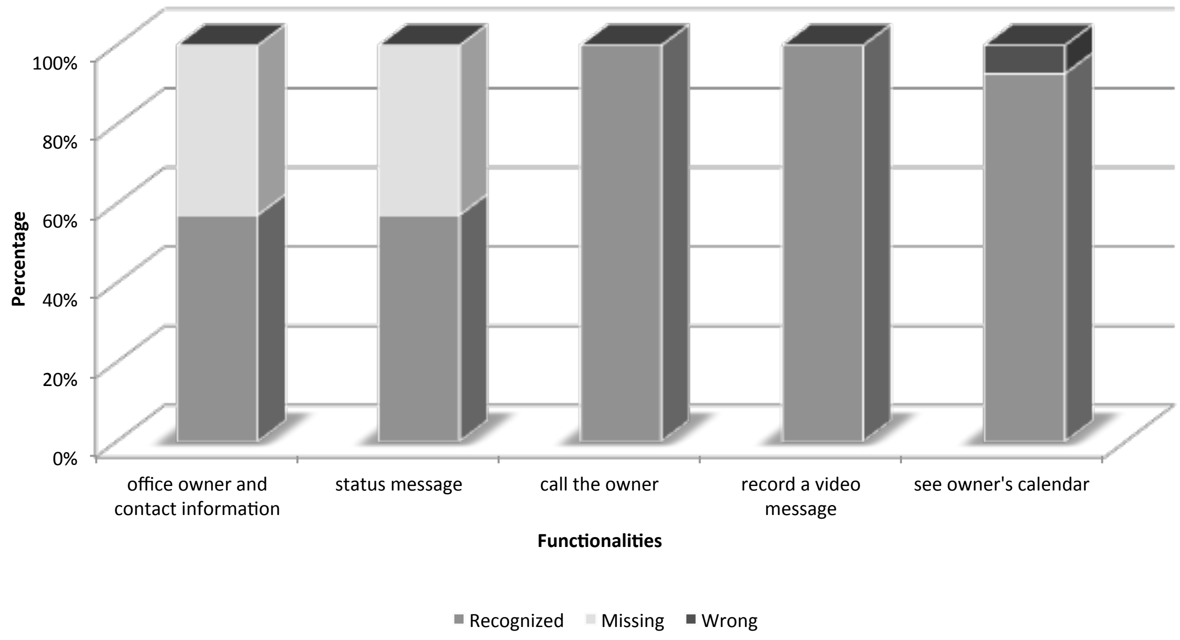


Figure 10.3: Identification of each functionality recognized by the participants (Part 1.2)

We can see from figure 10.3 that all the functionalities were recognized. However, the functionality *office owner and contact information* and *status message* were recognized only by 57% of the participants. Unlike the others functionalities, they didn't have a specific button. As for the two less recognized functionalities, they are obvious when the application is lanched, our conclusion leads us to believe that many participants took them for granted and did not mention them when answering question 2.

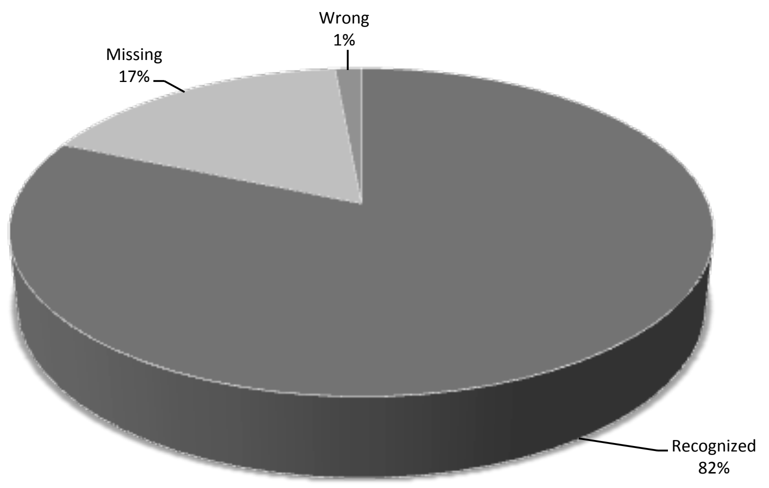


Figure 10.4: Percentage of the overall functionality recognized by the participants (Part 1.2)

As shown in figure 10.4, 82% of the overall functionality was recognized by the participants. This score is satisfactory given the success criteria described in section 10.5.6 (more than 80%).

Part 2 – Task questionnaire

In the second part, the participants were asked to complete a set of given tasks and then evaluate the difficulty of performing them. Figure 10.5 presents the results. For each task we can see how many participants evaluated it as *very easy/easy*, *intermediate* or *difficult/very difficult*.

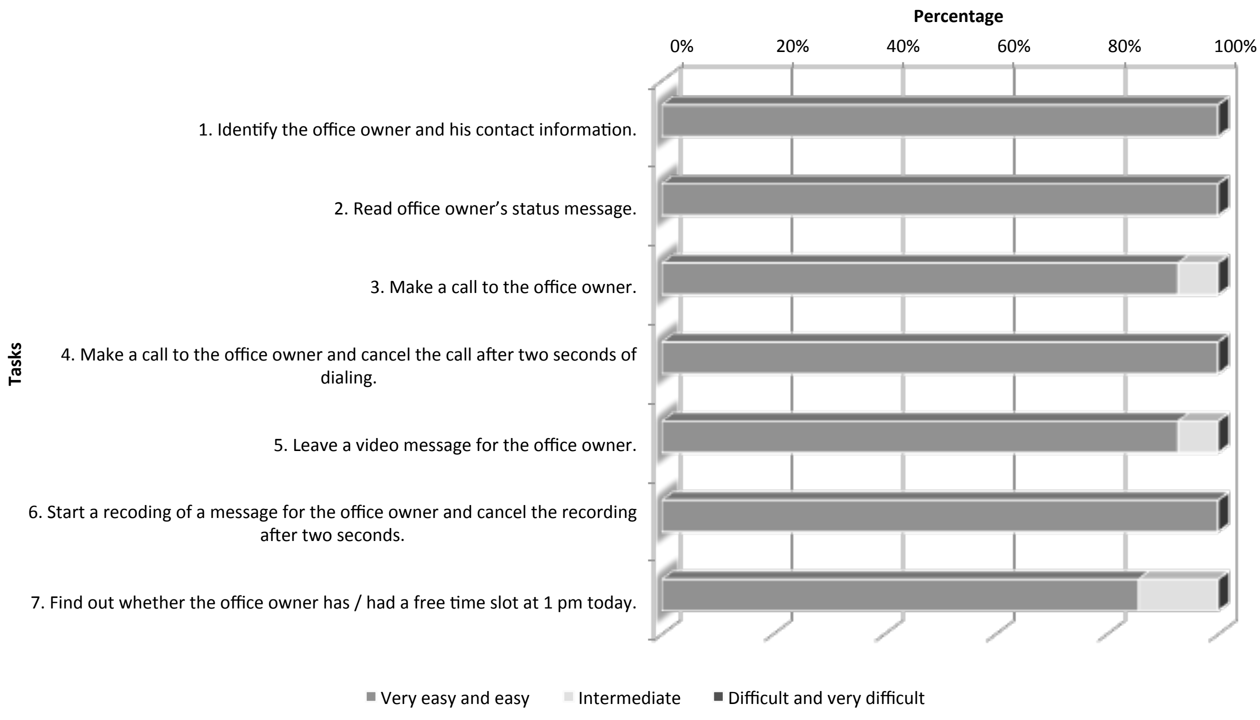


Figure 10.5: Difficulty level of tasks, as perceived by the participants.(Part 2)

From figure 10.5 we can see that each task was evaluated as being *very easy* or *easy* by more than 80% of the participants. These results are sufficient given the success criteria presented in section 10.5.6.

Part 3 – SUS questionnaire

The last part of the usability test was answering a System Usability Scale assessment [see section 10.3]. The figure 10.6 presents the results of this assessment and the distribution of the different scores obtained.

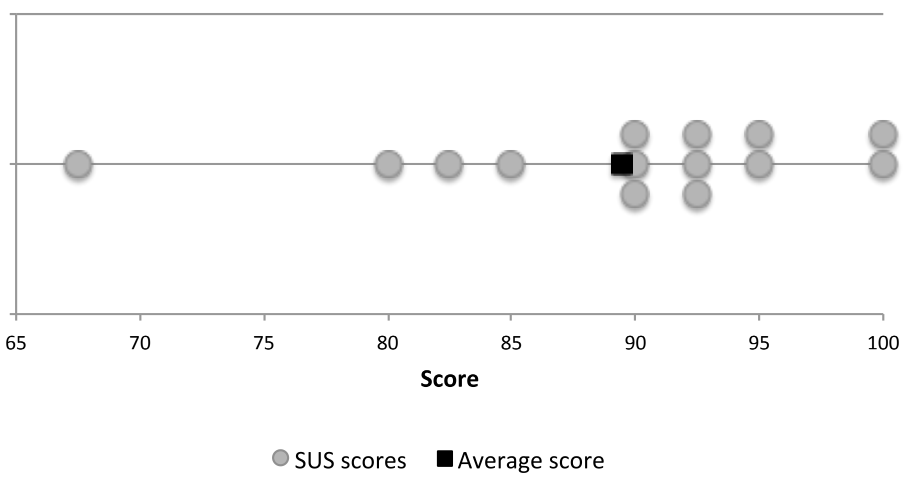


Figure 10.6: SUS evaluation (Part 3)

As we can see in figure 10.6, almost all the score are higher than 75 and the average is around 89.5. This score is excellent according to figure 10.1 and more than what was expected in section 10.5.6, which was a score higher than 70.

Summary

A summary of the results for the different parts of the questionnaire is shown in table 10.20.

Part, question	Success	Failure
Part 1, question 1		√
Part 1, question 2	√	
Part 2	√	
Part 3	√	

Table 10.20: Summary of results

Even though many of the participants did not fully understand the purpose of the system, most of them were able to recognize all the available functionality. Moreover, the participants rated most of the tasks given to them as *very easy* or *easy*, and the average SUS score is excellent. So, we can conclude that our application has good usability: It is easy to use, easy to learn and useful.

However, given the small amount of participants the accuracy of this usability test might be low. Given more time, we could have included more test subjects to complete the test, as well as using the results to improve the system.

10.7.2 Owner's perspective

Results

For the owner's perspective, only our customer participated. Firstly because of a lack of time and secondly because the most important part was the visitor's perspective. Since our application has integrated Skype and Google functionality, all of the interaction with the *DoorSign* will be done through either Skype or Google, this is part of the reason why we found this perspective less important than that of the visitor.

For this perspective the questionnaire was divided into two parts. In the first part the participant was asked to do some tasks and evaluate their difficulties after reading the user guide. The second part was a SUS assessment just like part 3 of the visitor's perspective [see section 10.3].

The figure 10.7 presents how our customer evaluated the difficulty of the different tasks in the owner's perspective test (part 1).

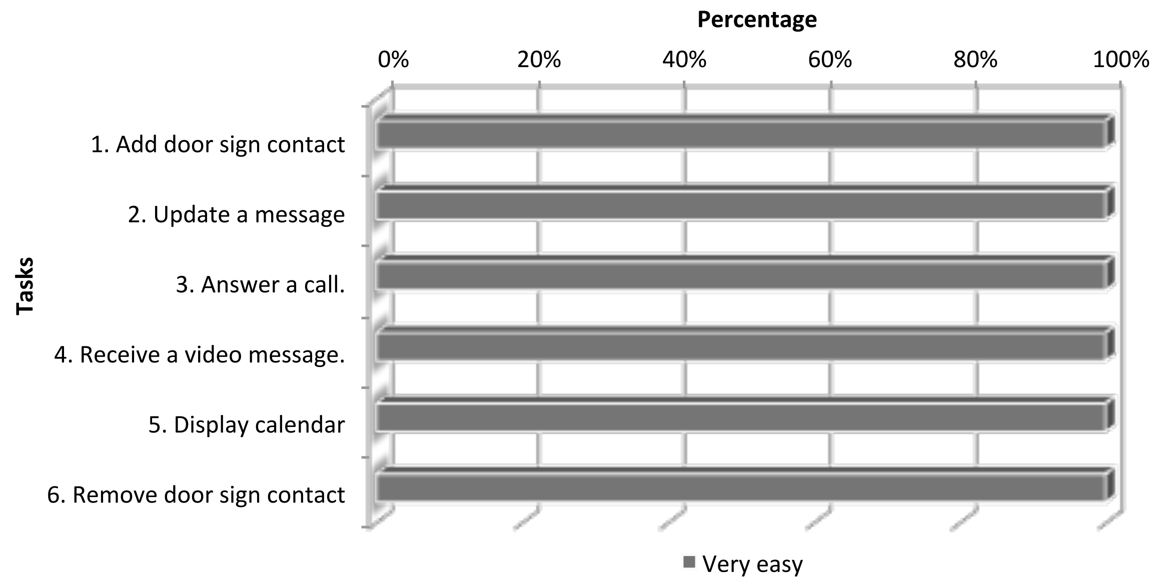


Figure 10.7: Difficulty level of tasks, as perceived by the participants. (Part 1)

As we can see in figure 10.7, our customer found all the given tasks very easy to perform. Even if one participant is not enough make any conclusions, we are very satisfied with this result (and it is more than what was expected in section 10.6.6).

For the SUS questionnaire, the score of our customer was 95, which is excellent according to figure 10.1.

Summary

One participant is clearly insufficient make any conclusions about the usability of our application for the owner's perspective, we are very satisfied with the feedback we got from our customer. Thus from our customer's perspective, the owner's perspective of our application has is very usable.

Chapter 11

Evaluation and discussion

In this chapter we will evaluate our process during the project to see what we did well and what we could have done better. We will start with an assessment of our planning and use of the Scrum methodology before we move on to our architecture. We defined some standards in our Quality assurance chapter that we will discuss here to see whether or not they were followed and how they worked out in general. To sum up this section, we will discuss our group dynamics on an overall basis, and give a recap of how we handled our risks and if our planned risk management was good enough.

In the next section of the chapter we will evaluate our results, meaning the fulfillment of our requirements, how our architecture worked and the results from our usability study. The latter will give great insight as to how good our product is. As a closing argument, we will discuss our planned effort versus our actual effort.

11.1 The process

11.1.1 Planning

During the project directive phase we created a project plan using Microsoft Project software. The original plan underwent four revisions during the project. These changes reflected new information gained during our progress and changes in our project. The most important circumstances that affected the project plan were:

- Delayed preliminary studies caused by device change – a lot of time was lost with the original device.
- Decision to postpone the usability study until after sprints and extend its scale.
- Delay at the end of the second (final) sprint.

The original project plan proved to be flexible enough to handle these issues. The plan contained appropriate buffers and enabled us, in spite of many changes to our project, to implement all required functionalities. The final version of our project plan can be seen in appendix C.

11.1.2 Scrum methodology

The decision to use the Scrum methodology was evaluated as good despite several problems with its usage. The core concept of Scrum – sprints – enabled us to divide our implementation work into independent logical parts. The first sprint focused on high-priority requirements and the second one on finishing remaining requirements. We decided to choose only two sprints as we wanted everyone from the development team to work on separate requirements and avoid moving from one task to another.

This decision determined our planning. During sprint planning meetings, requirements were estimated as a whole and were not broken down into smaller work packages. Smaller work packages would have probably led to more accurate estimations and made it necessary to go deeper into the design of particular application functionality.

During the self-study of the Scrum methodology and its processes, especially the sprint planning meeting, we obtained new information that the lecture about Scrum had not revealed. The new information was that during daily Scrum meetings, the small tasks making up the sprint backlog are dynamically assigned to team members who have finished their previous tasks. This is a completely different approach than what we did. From today's perspective, it would probably have been better.

Daily Scrum meetings were evaluated as a very beneficial tool to maintain awareness among the members of the development team. However, we could have benefited more from this process if we had used task assignment as described earlier. The same applies for the burn down chart – it was a beneficial tool for monitoring our progress, but smaller task would have made it more accurate.

In spite of these issues, we consider Scrum as one of the best methodologies for managing implementation of projects such as ours.

11.1.3 Quality assurance

External communication

Customer

The communication with our customer was done mainly during meetings arranged by our customer contact. These meetings were not regular but reflected our team's needs to consult important points and get agreement on the results of the sprints. The feedback we got from the customer after meetings, and especially after our demonstrations of our product, was very beneficial and helped us stay on the right track.

There were not any problems in the relationship and communication process with the customer despite the many changes in our project. At the beginning we had an agreement on the project charter and were able to create a shared view of the requirements specification. During the project we were able to handle a device change when preliminary studies showed that we would not be capable of meeting requirements with the original device. The final requirement change during the second sprint and negotiating about new functionality led to an acceptable solution for both sides.

Advisor

Although we had the opportunity to contact him through e-mail, communication with our advisor was held almost exclusively during our weekly advisory meetings on Thursdays. Our preparation for these meetings included a weekly report, which contained the agenda of the meeting and questions for the advisor, and the most recent version of our final report. Our questions were answered and feedback to particular parts of our work was given during these meetings.

On the other hand, we would appreciate a more consistent approach from our advisor in team dynamics matters. There were several personal discussions between team members and the advisor, but these were not reflected during general discussion with the whole team.

Our advisor gave us much needed feedback and gave the process a push if we seemed to be stuck. Several important topics were discussed and subsequent actions were taken after our weekly advisory meetings. Overall it was a useful hour we spent every week that helped ensure our project's quality, although at times we would have preferred longer meetings.

Internal communication

Ground rules

Our evaluation of the communication within the team is explained in greater detail in the Group dynamics section of this chapter [see section 11.1.4].

We only had a few ground rules to follow [see appendix E], and due to stress and lack of motivation towards the end of the project, some of these rules were forgotten.

The third rule was one of the only ones that got better as time went on. People were able to speak freely without being interrupted, and if someone interrupted another team member, they would apologize and let them finish. The more we got to know each other, the more we respected each other, of which this is clearly a reflection.

The language slipped a few times, but we mainly spoke English so that everyone would understand. As for deadlines, we ended up having only a few of them, thus making this rule easy to follow as nearly everything was finished on time. It was also one of the easiest ones to break. No deadlines meant we could decide how long we were going to work on our parts. We could perhaps have had more deadlines or milestones to help with this rule.

Not too many beers were bought as we did not keep track of who was late, and it was decided later on in the project that this was not a suitable punishment. It would have been enough for the project manager to simply have a private chat with those who for example were late to meetings or did not hand in their work on time.

Implementation

We followed the coding convention mentioned in the Planning and Quality assurance chapters [see section 6.1.2 and 2.7.2]. All class names, methods and variables are written in English and classes and variables are described using Javadoc. Thus, it was, as predicted, very easy to maintain the code and for others developers to understand it and be able to connect everything. Also, for further developers it will be quite easy to understand the code thanks to Javadoc and comments in the code.

As mentioned in section 2.1.2, we used SVN as a revision control system. It was a very good choice for us because we did not have any issues with it (not a lot of conflicts, no merging, etc.). So, it was an easy way to share the code between developers and to maintain the current version of it.

Documentation

Using Dropbox we got a full overview of the various sections of our final report. We ended up using only lower-case letters in our Latex-files, while the files in our other folders included capital letters. Most of the documents used underscores instead of spaces. The ones who did not were still functional and no one experienced any errors, and as we had made the standards after we had already created some of these documents, we decided against changing their names just so they would meet them. We made sure every file name was in English, and as descriptive as possible.

As the project grew, so did the number of folders in Dropbox. Each folder was structured so that navigation would be as easy as possible, and reflected the structure of the final report. Documents regarding the report were in separate folders than for example summaries of meetings. This was done so that they would not be confused with each other.

All references were generated using BibTeX, which made our job a lot easier.

Templates

Writing summaries of meetings and the weekly report went fast as we used our templates. It shaved off a lot of the time we would have spent setting up the documents and figuring out how to write them. All we had to do was follow their structure, with some minor modifications, and it would be finished within minutes.

The last part of the summaries that had information about future meetings could have been utilized by better planning. Had all meetings been planned before the end of the last, or had we decided to have regular meetings, as will be mentioned in the next section [see 11.1.4], it would have been easier to follow and regulate.

Our weekly reports helped set the agenda for the advisory meetings. This way we would get specific feedback on our work instead of vague comments we might have gotten if we had shown up without a plan for what we wanted to discuss with our advisor. The advisor also got a feel of our progress as our hours and work were presented at these meetings and in the report, and would therefore try to give us some extra motivation to work even harder.

Testing

A detailed evaluation of our testing has been presented at the end of our Usability study chapter [see section 10.7]. Here we have described the procedure of the testing and which standards we have had for our tests, which helped ensure the overall quality of our product, and evaluated the results.

11.1.4 Group dynamics

Despite different backgrounds of our team members our team worked well together and was able to handle the different situations that occurred during this extensive project. There were not any major conflicts or disagreements among team members, and minor problems were successfully solved.

The communication process in general could have been better. As it is discussed in next section, we had problems finding common time slots for meetings and this limited us. Another point, and great experience for the future, is a value of investment into the team. We did not set up any team building meetings that would have provided us with opportunities to get to know other team members better. If it had been done, it would probably have brought the team work to a higher level.

Our team also encountered troubles with team members being late, as mentioned earlier in this chapter. We set up ground rules which should have handled this and similar matters, but their concept was not elaborated enough. The project manager should have paid more attention to them, repeated them on regular basis and enforced them. Another useful operation would probably be elaboration and promotion of the list of ground rules to the *group contract*. This contract would be signed by all team members and obligated them to follow them.

Communication within the team

At the very beginning of the project we found out that arranging regular meetings would be problematic due to the fact that each team member had a different schedule. This became even more clear when we used the web service *Doodle* to compare our schedules. Our difficulties arranging common meetings during our project were intensified in later phases by travel activities, illness, conflicting project deadlines and other issues. We were only able to establish one common team meeting on Tuesdays and one advisor meeting on Thursdays from the beginning of the project.

As a result we had to arrange many ad hoc meetings which were attended by only a part of our team and did not have reserved rooms. This affected mainly our usage of the agile Scrum methodology, especially daily Scrum meetings, and led to an overload of cooperation activities.

The software planned for communication and cooperation worked fine during the project. We were able to handle problems that occurred due to a lack of regular meetings. One exception was Internet Relay Chat (IRC), which was used rarely because its console GUI was not user-friendly. Skype and e-mails were adequate replacements.

We usually used the *reply to all* function when we sent e-mails about the project, which made it easier for the whole team to stay updated, but we could have done it more consistently. One reason why it did not work 100% of the time could have been because we failed to include it in our ground rules. We also could have set a deadline so that no e-mails would be sent after a certain time, for example 6 PM, eliminating uncertainty as to whether or not there was going to be a meeting the following day. The way we did it, we would sometimes get e-mails only a few hours before meetings. We would also have had more time to prepare to meetings if we had known about them earlier.

With today's experience we would have set up more regular meetings, preferably four, and reserved rooms for them in spite of the fact that they could only be attended by a small part of the team. We would probably set up a ground rule obligating a team member to attend at least two or three of them, with some allowed exceptions. This would increase our effectiveness and decrease the amount of cooperation activities. We would also have set up some extra rules about e-mailing so that there would be as little confusion as possible about meetings while everyone would be fully updated.

Roles

Role and work distribution was a strong part of our team. Our team had three international students and was very heterogeneous – almost all team members had different backgrounds. This diversity was clarified at the beginning of the project, and we benefitted from it and utilize the strong sides of particular team members. Based on knowledge of our strong sides we were able to divide work so that team members did what they were good. We also gave everyone the opportunity to work on various tasks strictly to improve their skills.

Of course, there were many cases where this was not applicable and team members had to do other tasks that were necessary or tasks where they were not so skilled. But in general, roles and responsibilities were stated well and influenced our project in a very good way.

Distribution of work

A big issue in our group was balancing the workload. Our team did not work as a self organizing structure where everyone was committed to reaching the required amount of working hours. Filling in the workload table required for weekly reports did not come naturally during the progress of our project and had to be reminded. It was mainly seen as a nuisance and a distraction.

We experienced that it was difficult to reach 25 working hours per week and that to replace a week's absence is almost impossible in a close time frame. A team member had an excused week long absence in the first week and this could cause trouble in commitments to compare the workload to other team members.

The project manager fulfilled all required work by delegating task, but reaching a balanced workload of all team members using this approach is very difficult, especially when there are planned traveling activities, unpredicted illnesses, conflicting projects and many other similar issues.

11.1.5 Risk management

In this section we will discuss the risk identified during the risk analysis [see section 2.6] which were encountered and the way we handled them.

- Problems with hardware – The original device caused many problems during our preliminary studies. We spent a lot of time trying to update the operating system in order to be able to fulfill requirements. In the end we had to reject the device and do some parts of the preliminary studies again for the new device. The result was a delay which was handled using a buffer in our plan and a reduction of the time dedicated to architecture studies.
- Problem with implementation – We encountered problems with implementation, especially during the second sprint. We were not able to implement the video part of the video conference feature and had to negotiate a requirement change. Another problem was with Java library which did not detect our device. This resulted in problematic synchronization of audio and video that was recorded separately using a different library. These problems resulted in several days' delay of the second sprint. As a result, not all team members could work on the usability study as was intended.
- Interfering work / activity of a team member – This was a very common problem within all phases of the project. Problems caused by this risk were handled by keeping minutes from all meetings, which kept team members updated, and delegation of work to other team members.
- Illness – This was a quite common problem causing trouble during all phases of the project. Problems caused by this risk were handled by keeping minutes from all meetings, which kept team members updated, and delegation of work to other team members.
- Lack of communication internally – We encountered this problem during several days of the second sprint as a result of the *interfering work / activity of a team member* risk described above. This problem contributed to delay caused by the *problem with implementation* risk described above.

From the described risks we can see the way they have been solved – using buffers in the plan, by reducing time dedicated to particular phases or by delegating work to other team members. Our risk analysis was precise and we did not encounter any new risks during our progress.

11.2 The result

11.2.1 Requirements fulfillment

We were able to implement all final requirements. One original requirement – F5 (video conference) was implemented partially, meaning only the audio part was available, and was replaced by the new requirement F8 (calendar) [see section 9.5.2]. A short summary of our requirement specification can be seen in table 11.1.

Requirement F5 together with other potential improvements are discussed in section 12.4.

Quality requirement Q1 (Security) was fulfilled by the security analysis described in section 9.3 and requirement Q2 (Usability) by our usability study described in section 10. The last quality

#	Short description	Priority	Finished	Finished in
F1	Displaying Information	H	√	sprint 1
F2	Update message remotely	H	√	sprint 1
F3	Record video message	M	√	sprint 2
F4	Store video message	M	√	sprint 2
F5	Video conference	M/H	×	–
F6	Database	H	√	sprint 1
F7	Send video messages	L	√	sprint 2
F8	Calendar	M	√	sprint 2
Q1	Security	M/H	√	sprint 2
Q2	Usability	M/H	√	after implementation
Q3	Performance	M	√	sprint 2

Table 11.1: Requirement fulfillment

requirement – Q3 (Performance) means *to be able to have a normal conversation through the video conference*. As we used the Skype platform for the audio part of the video conference, we were not able to influence the performance of the call in any way.

11.2.2 Architecture

One of the most important architectural decisions we made was to use the Model View Controller (MVC) pattern. This pattern was used to comply with the quality requirement for the usability of the final product.

Throughout the implementation process we kept to this pattern, only expanding the number of classes used in the packages. The resulting class diagrams 5.7.1 looks a lot bigger compared to diagram 5.1 planned in the architectural stage. The amount of classes used in packages grew mostly because the team was not familiar with the software solutions used for this project (SkypeKit, JMF). Most of the class and method names used in the architectural stage were kept intact. Some of the class and method names were changed, because as the team got more comfortable with used software solutions, more appropriate names were chosen.

Considering the size of the application and the requirements, we consider the architectural decisions made to be sufficient enough.

11.2.3 Usability study

The usability study [see chapter 10] allowed us to measure how easy our application is to use from the owner and visitor's perspectives. This study was a big issue and took much more time than was originally expected because no one in our team had previous experience with it.

Nevertheless the results of this study for the visitor's perspective [see section 10.7.1] were very good because most of the participants figured out the functionalities without any prior knowledge of the application. Furthermore, when they were asked to realize some tasks, they found most of them to be *very easy* or *easy*, still without any prior knowledge.

Even if these results are very good they can still be improved [see section 12.4]. An example can be the calendar. Its GUI could have been more elaborated. This study also allowed us to get some feedback from end users especially about the user interface. For example, some participants thought that some of the names of the buttons could be improved, such as *Place call*. Some participants were also wondering who they were actually calling and how (via Skype, cell phone, etc).

Even if we did not get the chance to have more participants for the owner's perspective [see section 10.7.2] we were very satisfied with the answers given by our customer, as he thought the application was a success.

11.2.4 Planned effort

As written in section 2.1.1 our estimated workload was 325 person-hours in 13 weeks which means 2275 working hours in total. Our final workload was 1913 working hours, which is a considerably lower number.

During our project we found it difficult to reach the expected amount of working hours per person per week, which was 25. The reason was absences, often up to a week, due to traveling activities and illnesses, among other things, resulting in the necessity to work 25 additional hours during the next several weeks. This showed to be unrealistic. The same applies to conflicting projects and other work.

On the other hand, at the end of the project we were able to provide a fully functional application with all requirements fulfilled, even though our workload was below the expected amount.

The picture 11.1 shows our weekly workload. Work done before the first advisory meeting when our effort registration began was added to week 36. The chart reflects our progress. During week 38 and 39 we faced problems with our original device and the device limited simultaneous work. This caused delay and prolonged the preliminary studies and so we had only one week (week 40) to finish our architecture studies.

The next four weeks were dedicated to implementing the system, and during this period, together with week 45, we faced problems with our workload, as described earlier in this section. During week 45 we prepared a plan for our usability study and executed it. The last two weeks were spent mostly on finalizing the report and making a presentation.

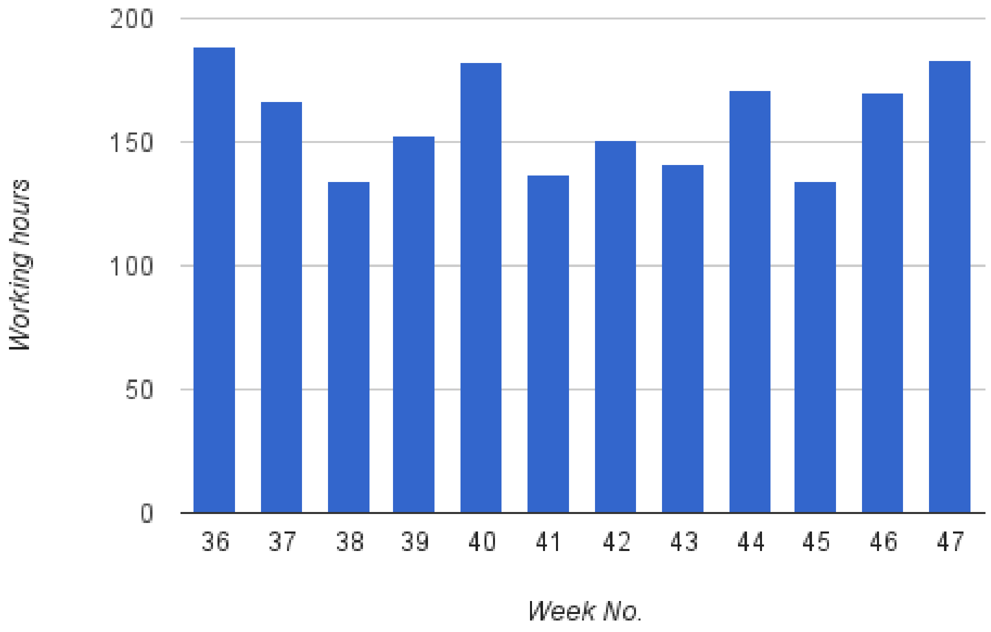


Figure 11.1: Weekly workload

Chapter 12

Conclusion

This chapter serves as a summary of our project. We will answer two questions; whether or not we managed to create the product our customer wanted, and whether or not we accomplished our own goals and expectations.

We will end the chapter with an explanation of the future development of our product by giving a rundown of its limitations and the potential future work that can be done.

12.1 Requirements and expectations of the customer

The main requirement of our customer was a working prototype of the door sign application. The detailed descriptions of the requirements can be found in chapter 4. There were also two major quality requirements to focus on – the usability and security of the application.

Our customer was set on using the Skype platform as a solution for the office owner's part, which we had in mind when choosing our solution. Towards the end of our second – and final – sprint, we had created a fully functional application with all *final* requirements implemented.

A Java application was created for the client part, and includes most of the original requirements, except one. The video conference feature was not possible to implement, so instead we implemented a calendar enabling the office owner to share their Google calendar and make it visible to visitors. This also adds as an extra clue as to whether or not the office owner is available. The evaluation of our software security, described in sprint #2, ensured the security of the application.

To make sure our product fulfilled the requirements and expectations of our customer, we had two demonstrations – one after each sprint. Our customer expressed that he was happy with what we had done, and that the product design was even better than he had expected. Based on this information, we have come to the conclusion that the product we created met the customer's expectations and we are therefore classifying it as a success.

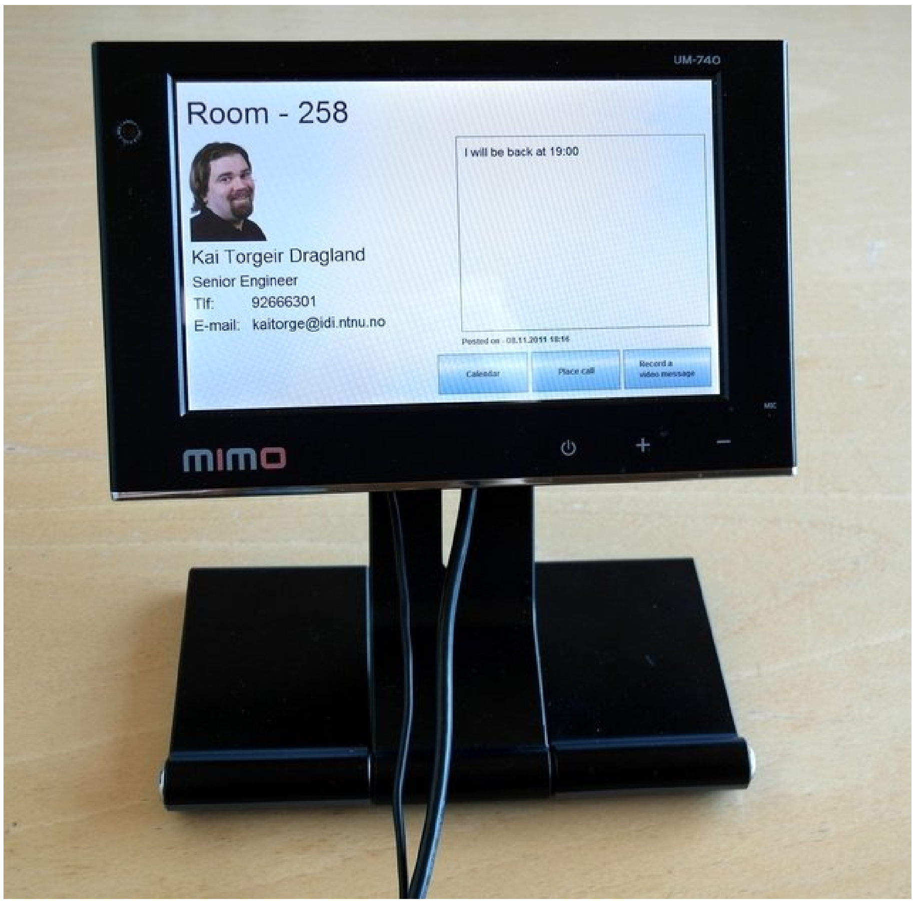


Figure 12.1: The final product

12.2 Goals and expectations of our team

At the beginning of our project we wrote down the goals and expectations of our team. First and foremost we wanted to develop a functional prototype fulfilling the requirements provided by our customer together with accurate documentation. The second goal was to learn more about working with team members with different backgrounds and different levels of experience.

During one of our first group meetings we collected expectations from every team member. Below is a list of these expectations and a check mark showing whether or not we feel they were fulfilled.

Expectations	**Fulfilled**
Learn enough from the experience to feel we can use it in future projects	√
Work as a team, regardless of our different backgrounds and international students	√
Understand and do what the customer wants	√
Generate a good product that the customer likes	√
Make a working prototype	√
Get a good group dynamic and manage to work well together without too many problems	√

12.3 Limitations of the product

Software limitations

- The application was created and tested on a Windows platform.
- To successfully use the application with the device, the user has to have a Skype account. The application does not support any other means of message updating and voice calls.
- A video recorded message cannot be longer than 30 seconds.

Hardware limitations:

- While being touched, the touch screen goes into a loop of clicks that creates navigational problems. There is no way to change settings defining interaction with the touch screen.

12.4 Future work

In this section we will describe the various things that can be developed further or added to our product.

To attract a wider range of users, the application could be adjusted to work on different platforms other than Windows, and have more than one way to interact with the device through internet to change the status message or receive calls.

As mentioned in earlier chapters and the first section of this chapter, the requirement we did not implement was the video conference feature. The application functionality can be expanded to include a video conference between the office owner and visitors. This will be possible as soon as Skype releases the new version of SkypeKit that supports video conferencing.

Video recordings can be improved to capture video recordings with a higher quality and length if a better video capture device that is supported by JMF is used.

The design of the calendar could be improved, and the calendar itself could have more functionality, for example letting people interact with it. This could include viewing another day's schedule or requesting an appointment.

The application could do with better exception handling. At this point most of the errors caused by external factors (e.g. Skype, Google Calendar, MySQL) will cause the application to crash.

During the usability study, we received a lot of useful feedback from potential users of the device. We got suggestions from the test subjects and our advisor regarding additional features. When or if these have been implemented, a new usability study can be conducted. The new usability could then be greater in scale as the one we conducted was limited due to our project's scale.

Bibliography

[1] Adobe. Adobe livecycle collaboration service. http://www.adobe.com/products/livecycle/collaborationservice/, 2011. Visited: 19. 09. 2011.

[2] Angrybirds.com. Angry Birds [online]. http://www.angrybirds.com, 2011. Visited: 12. 09. 2011.

[3] Aaron Bangor, Philip Kortum, and James A. Miller. Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale. http://www.usabilityprofessionals.org/upa_publications/jus/2009may/JUS_Bangor_May2009.pdf, May 2009. Visited: 04.11.2011.

[4] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Addison Wesley, second edition edition, 2003.

[5] Mike Beedle, Arie van Bennekum, Ward Cunningham, Martin Fowler, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Ken Schwaber, Jeff Sutherland, and Dave Thomas. Manifesto for agile software development. http://agilemanifesto.org/iso/en, 2001. Visited: 10.09.2011.

[6] John Brooke. SUS – A quick and dirty usability scale. http://hell.meiert.org/core/pdf/sus.pdf. Visited: 04.11.2011.

[7] Wittawat Chaisaraseree. Scrum methodology with software development works. http://software-document.blogspot.com/2011/07/scrum-methodology-with-software.html. Visited: 28.10.2011.

[8] Product Manager Colin Gibbs and Tech Lead Wei Huang. Video chat on your android phone. http://googlemobile.blogspot.com/2011/04/video-chat-on-your-android-phone.html, 28.04.2011. Visited: 20.09.2011.

[9] Developer.skype.com. Skypekitoverview. https://developer.skype.com/images/SkypeKitOverview.png. Visited: 28.10.2011.

[10] Developer.skype.com. SkypeKit [online]. http://www.developer.skype.com, 2011. Visited: 12. 09. 2011.

[11] Eclipse. The eclipse foundation open source community website. http://www.eclipse.org/, 2011. Visited: 21. 09. 2011.

[12] Freewarelovers.com. PacMan [online]. http://www.freewarelovers.com/android/app/pac-man, 2011. Visited: 12. 09. 2011.

[13] Google. Open communication - google talk for developers. http://code.google.com/intl/en/apis/talk/open_communications.html, 2011. Visited: 17. 09. 2011.

[14] HardcoreHacker. Uberoid universal honeycombmod. http://techknow.freeforums.org/viewforum.php?f=112&start=&sid=4c263d0467b63d7178ec586bfff14efc, 2011. Visited: 21. 09. 2011.

[15] Lti-civil.org. CIVIL - Capturing Images and Video In a Library. [online]. http://lti-civil.org, 2011. Visited: 27. 10. 2011.

[16] Microsoft. MSDN Academic Alliance [online]. http://msdn.microsoft.com/en-us/academic/bb250591, 2011. Visited: 07. 09. 2011.

[17] Mimomonitors.com. Mimo Support site [online]. http://www.mimomonitors.com/pages/customer-support, 2011. Visited: 28. 10. 2011.

[18] Dong Ngo. Mimo um-740 monitor. http://reviews.cnet.com/displays/mimo-um-740-monitor/4505-3174_7-33513709.html. Visited: 15.11.2011.

[19] Oracle. Documentation for class Socket Java. http://download.oracle.com/javase/1.4.2/docs/api/java/net/Socket.html, 2011. Visited: 21. 09. 2011.

[20] Oracle.com. Sun Oracle's Java Media Framework [online]. http://www.oracle.com/technetwork/java/javase/tech/index-jsp-140239.html, 2011. Visited: 23. 10. 2011.

[21] OWASP. Microsoft Threat Modeling Process [online]. https://www.owasp.org/index.php/Threat_Risk_Modeling, 2011. Visited: 10. 11. 2011.

[22] OWASP. OWASP Risk Rating Methodology [online]. https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology, 2011. Visited: 10. 11. 2011.

[23] Winston W. Royce. Managing the development of large software systems. http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf, 1970. Visited: 10.09.2011.

[24] Jeff Sauro. Measuring Usability with the System Usability Scale (SUS). http://www.measuringusability.com/sus.php, February 2011. Visited: 04.11.2011.

[25] Skype.com. Skype App [online]. http://www.skype.com/m, 2011. Visited: 12. 09. 2011.

[26] Peter Stevens. A simple scrum sprint review. http://agilesoftwaredevelopment.com/blog/peterstev/simple-scrum-sprint-review, 01.21.2009. Visited: 13.09.2011.

[27] Usability.gov. Usability test plan template. http://www.usability.gov/templates/docs/u-test_plan_template.doc. Visited: 30.10.2011.

[28] waterfall model.com. Waterfall model - all about the waterfall model. http://www.waterfall-model.com/. Visited: 28.10.2011.

[29] Wikipedia. Scrum (development). http://en.wikipedia.org/wiki/Scrum_(development). Visited: 02.11.2011.

[30] Wikipedia. Usability. http://en.wikipedia.org/wiki/Usability, 04.11.2011. Visited: 09.11.2011.

[31] Wikipedia. Waterfall model - supporting arguments. http://en.wikipedia.org/wiki/Waterfall_model#Supporting_arguments, 06.05.2011. Visited: 10.09.2011.

[32] Wikipedia. Software testing. http://en.wikipedia.org/wiki/Software_testing, 16.09.2011. Visited: 28.09.2011.

[33] Wikipedia. Java's garbage collection [online]. http://en.wikipedia.org/wiki/Garbage_collection_%28computer_science%29, 2011. Visited: 27. 10. 2011.

[34] Xuggle.com. Xuggler [online]. http://www.xuggle.com/xuggler/, 2011. Visited: 27. 10. 2011.

Appendix A

User and installation guide

A.1 Installation guide

1. Mimo Drivers
   The links below will guide to to a site where you must choose different driver for XP, Vista, Mac and Windows 7
   Follow the instructions on the webpage and installation files.
   beginenumerate
2. Display driver: http://www.mimomonitors.com/pages/customer-support
3. Touchscreen driver: http://www.mimomonitors.com/pages/customer-support endenumerate
4. The newest Java JKD (Java 7)
   - Download link: http://www.oracle.com/technetwork/java/javase/downloads/jdk-7u1-download-513651.html
5. Xuggler
   A third-party library for Java you need to record and store video.
   Follow the instructions on the webpage and installation files.

   -Download link: http://www.xuggle.com/xuggler/downloads/
   A bit down on the page you will find 'xuggle-xuggler-setup.exe', it says it's for XP and Vista, but it works with Windows 7 as well.
6. Restart you computer, this is necessary.

After the 4 steps above is completed, you should be able to run the Java application. Please proceed to following way:

1. Plug in the Mimo mintor
   (a) Configure the display setup to your preference via the installed Mimo display driver.
2. Open the folder, USB or DVD given by Group 7.
   (a) Under /Compiled_Code folder run the file *run.bat*.

(b) Follow the instructions given.

A.2 User guide

There is two actors for the door sign. We will provide a user guide for each actor. The first actor is the *Office owner*. He is the owner of the office where the door sign is located. The second actor is a *Visitor*. A *Visitor* is a person in front of the door sign who wants to use it.

A.2.1 Office owner

The different functionalities for the office owner are the following :

1. Add door sign contact
2. Remove door sign contact
3. Update message
4. Answer a call
5. Receive video message
6. Display calendar

A description for each functionality is given below.

Add door sign contact

1. Press *Add contact* in Skype
2. Find the proper Skype name of the door sign
3. Enter the password of the door sign Skype account as the request message
4. Press *Send request*

Remove door sign contact

1. Open a conversation with the door sign Skype account
2. Send the door sign account's password as an *Instant Message*
3. You would then be removed from the door sign contact list

Update message

1. Open a conversation with the door sign Skype account
2. Enter your personal message in the text field
3. Send the message and wait for a reply from the door sign that it has been updated

Editing the last message you sent to the device will also edit the personal message on the door sign.

Answer a call

Answer calls from the door sign account as you would for any other Skype call.

Receive video message

The door sign account will send you a video file if a video message has been left by a user of the door sign. You can receive this message the same way you would receive any other file using Skype.

Display calendar

You can display today's events from your Google calendar on the door sign by sharing your Google calendar with the Google account of the door sign.

A.2.2 Visitor

The different functionalities for the visitor are the following :

1. View the calendar
2. Place a call
3. Record a video message

A description for each functionality is given below.

View the calendar

To view the calendar, press the button *Calendar* on the main screen. This will show a calendar of today's events from the owner of the office Google calendar.
Press *Cancel* to return to the main screen. The door sign will automatically go back to the main screen after 1 minute.

Place a call

To call the owner of the office, press the button *Call.* You can cancel the call and return to the main screen by pressing the button *Cancel.* If the owner does not answer or if he rejects your call, a message indicating that the owner is not available will be displayed. From this new screen, you can return to the main screen by pressing the button *Cancel* or record a video message by pressing the button *Record a video message.*

Record a video message

1. Press *Record a video message*
2. Press *Start recording*
3. Now, you have one minute to record your message.
4. When you are done, press *Send* to send your recording to the office of the owner and return to the main screen.

At any time, you can cancel the record and return to the main screen by pressing *Cancel* (the recording, if started, will not be sent to the owner of the office).

If you do not press the button *Send* before the end of the minute, your recording will be stopped and not send to the owner of the office. You would still return to the main screen.

Appendix B

Templates

B.1 Group meeting

Date:
Time:
Place:

Attendees:

Andreas Nordahl	☐
David Andrássy Eilertsen	☐
Elise Boinnot	☐
Igoris Tirmailovas	☐
Knut Esten Melandsø Nekså	☐
Michal Krajíček	☐
Sibel Ayper Taşdemir	☐

Agenda:

Item	**Topic**	**Time (minutes)**
1		
2		
3		
4		
5		

Next meeting scheduled for:

Notes

B.2 Customer meeting

Date:
Time:
Place:

Customer:

Attendees:

Andreas Nordahl	☐
David Andrássy Eilertsen	☐
Elise Boinnot	☐
Igoris Tirmailovas	☐
Knut Esten Melandsø Nekså	☐
Michal Krajíček	☐
Sibel Ayper Taşdemir	☐

Agenda:

Item	Topic	Time (minutes)
1		
2		
3		
4		
5		

Next meeting scheduled for:

Notes

B.3 Weekly report

Group no.:
Week no.:
Date:

Attendees:

Andreas Nordahl	☐
David Andrássy Eilertsen	☐
Elise Boinnot	☐
Igoris Tirmailovas	☐
Knut Esten Melandsø Nekså	☐
Michal Krajíček	☐
Sibel Ayper Taşdemir	☐

Agenda of next advisor meeting:

Item	Topic	Time (minutes)
1		
2		
3		
4		
5		

Status report:

Problems and risks:

Summary of cutomer meeting:

Summary of last advisor meeting:

Plan for the next week:

Other:

Week workload

Team member	Meeting	Lecture	Individual	Group	Total
Andreas Nordahl					
David Andrássy Eilertsen					
Elise Boinnot					
Igoris Tirmailovas					
Knut Esten Melandsø Nekså					
Michal Krajíček					
Sibel Ayper Taşdemir					
Week total (hours)					

Individual work description

Team member	Description
Andreas Nordahl	
David Andrássy Eilertsen	
Elise Boinnot	
Igoris Tirmailovas	
Knut Esten Melandsø Nekså	
Michal Krajíček	
Sibel Ayper Taşdemir	

Appendix C

Gantt chart

ID	% Complete	Task Name	Duration	Start	Finish	Predecessors
1	**100%**	**Customer Driven Project**	**62,88 dys**	**Tue 30.8.11**	**Thu 24.11.11**	
2	100%	Lectures and self study	61 dys	Tue 30.8.11	Wed 23.11.11	
3	100%	Documentation	61 dys	Tue 30.8.11	Wed 23.11.11	
4	100%	Project management	61 dys	Tue 30.8.11	Wed 23.11.11	
5	**100%**	**Project directive**	**18,88 dys**	**Tue 30.8.11**	**Fri 23.9.11**	
6	100%	Project kick off	1 dy	Tue 30.8.11	Wed 31.8.11	
7	100%	Team organization	1 dy	Wed 31.8.11	Thu 1.9.11	6
8	100%	Cooperation technology	6 hrs	Thu 1.9.11	Thu 1.9.11	7
9	100%	Templates and standards	1 dy	Thu 1.9.11	Fri 2.9.11	8
10	**100%**	**Project planning**	**15,13 dys**	**Fri 2.9.11**	**Fri 23.9.11**	
11	100%	WBS	4 dys	Fri 2.9.11	Thu 8.9.11	9
12	100%	Relationships among tasks	2 dys	Fri 9.9.11	Mon 12.9.11	11
13	100%	Time estimations	4 dys	Tue 13.9.11	Fri 16.9.11	12
14	100%	Tests plans	3 dys	Sat 17.9.11	Wed 21.9.11	13
15	100%	Final plan creation	2 dys	Thu 22.9.11	Fri 23.9.11	14
16	**100%**	**Pre studies**	**14,93 dys**	**Fri 2.9.11**	**Fri 23.9.11**	
17	100%	Tablet analysis	10 dys	Fri 2.9.11	Fri 16.9.11	9
18	100%	Possible solutions	12 dys	Fri 2.9.11	Tue 20.9.11	9
19	100%	Development strategy	1 dy	Tue 20.9.11	Wed 21.9.11	18;17
20	100%	Security issues	1,93 dys	Wed 21.9.11	Fri 23.9.11	19
21	**100%**	**Requirements specifications**	**16 dys**	**Fri 2.9.11**	**Mon 26.9.11**	
22	100%	Requirements gathering	5 dys	Fri 2.9.11	Fri 9.9.11	9
23	100%	Prioritization	2 dys	Fri 9.9.11	Tue 13.9.11	22
24	100%	Use case creation	8 dys	Tue 13.9.11	Fri 23.9.11	23
25	100%	Risk analysis	7 dys	Fri 9.9.11	Tue 20.9.11	22
26	100%	Requirements approval	1 dy	Fri 23.9.11	Mon 26.9.11	24;25
27	100%	Introduction & planning phase completed	0 dys	Mon 26.9.11	Mon 26.9.11	26;20;15
28	100%	**Implementation**	**24,13 dys**	**Mon 26.9.11**	**Fri 28.10.11**	
29	100%	**Architecture**	**5 dys**	**Mon 26.9.11**	**Mon 3.10.11**	27
30	100%	**Sprint #1**	**9 dys**	**Mon 3.10.11**	**Fri 14.10.11**	
31	100%	Sprint planning	1 dy	Mon 3.10.11	Tue 4.10.11	29
32	100%	F1 (displaying information)	8 dys	Tue 4.10.11	Fri 14.10.11	31
33	100%	F2 (update message remotely)	8 dys	Tue 4.10.11	Fri 14.10.11	31
34	100%	F6 (database)	8 dys	Tue 4.10.11	Fri 14.10.11	31
35	100%	Q2 (usability)	8 dys	Tue 4.10.11	Fri 14.10.11	31
36	100%	Sprint #1 completed	0 dys	Fri 14.10.11	Fri 14.10.11	32;33;34;35

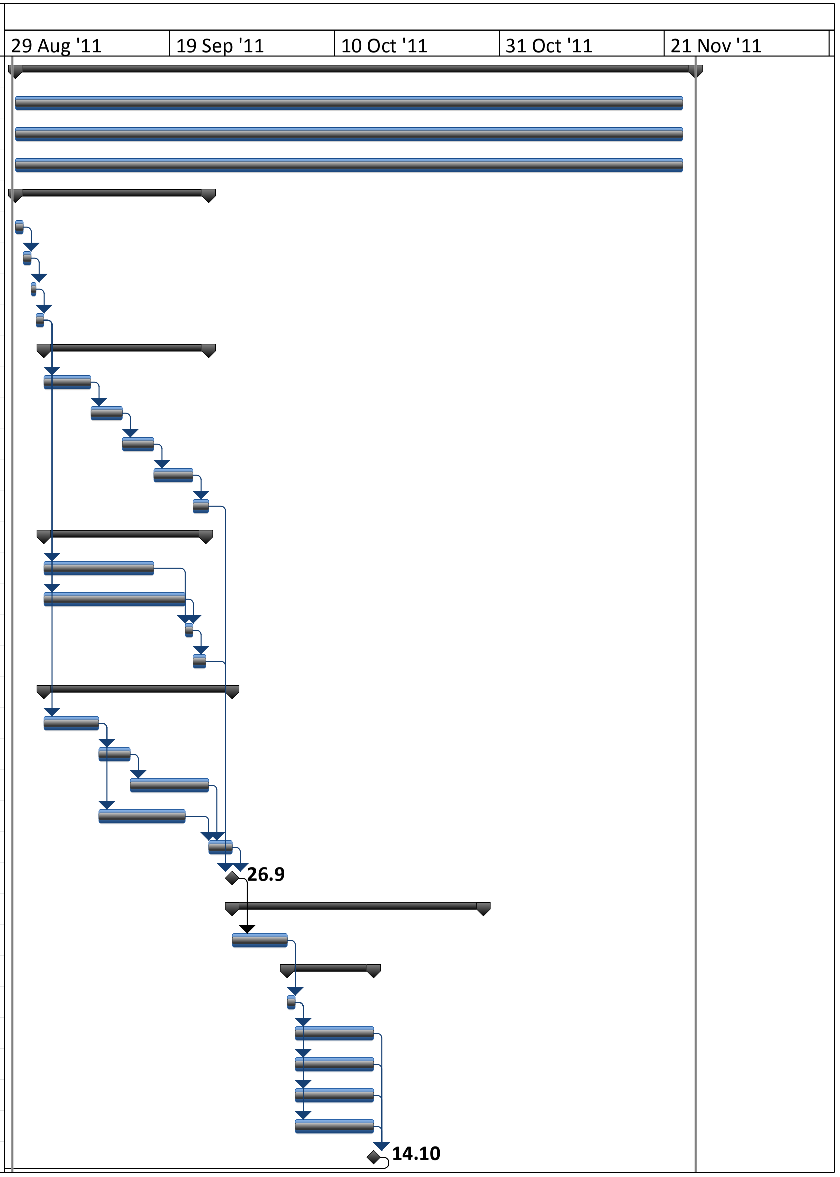


Figure C.1: Gantt chart 1/2

ID	% Complete	Task Name	Duration	Start	Finish	Predecessors
37	100%	**Sprint #2**	10 dys	Mon 17.10.11	Fri 28.10.11	
38	100%	Sprint planning	1 dy	Mon 17.10.11	Mon 17.10.11	36
39	100%	F3 (recording video)	9 dys	Tue 18.10.11	Fri 28.10.11	38
40	100%	F4 (storing video)	9 dys	Tue 18.10.11	Fri 28.10.11	38
41	100%	F5 (video conference)	9 dys	Tue 18.10.11	Fri 28.10.11	38
42	100%	F7 (downloading messages)	9 dys	Tue 18.10.11	Fri 28.10.11	38
43	100%	Q3 (performance)	9 dys	Tue 18.10.11	Fri 28.10.11	38
44	100%	Q1 (security)	9 dys	Tue 18.10.11	Fri 28.10.11	38
45	100%	Implementation completed	0 dys	Fri 28.10.11	Fri 28.10.11	39;40;41;42;43
46	100%	Usability study	5 dys	Mon 31.10.11	Fri 4.11.11	45
47	100%	Printer reservation	1 hr	Wed 2.11.11	Wed 2.11.11	45
48	100%	Creating final documentation	5 dys	Mon 7.11.11	Fri 11.11.11	46
49	100%	Documentation finished	0 dys	Fri 11.11.11	Fri 11.11.11	48;47
50	100%	Presentation creation	2 dys	Mon 14.11.11	Tue 15.11.11	49
51	100%	Project evaluation & conclusion	6 dys	Wed 16.11.11	Wed 23.11.11	50
52	100%	Project presentaion	1 dy	Thu 24.11.11	Thu 24.11.11	51
53	100%	Project finished	0 dys	Thu 24.11.11	Thu 24.11.11	52

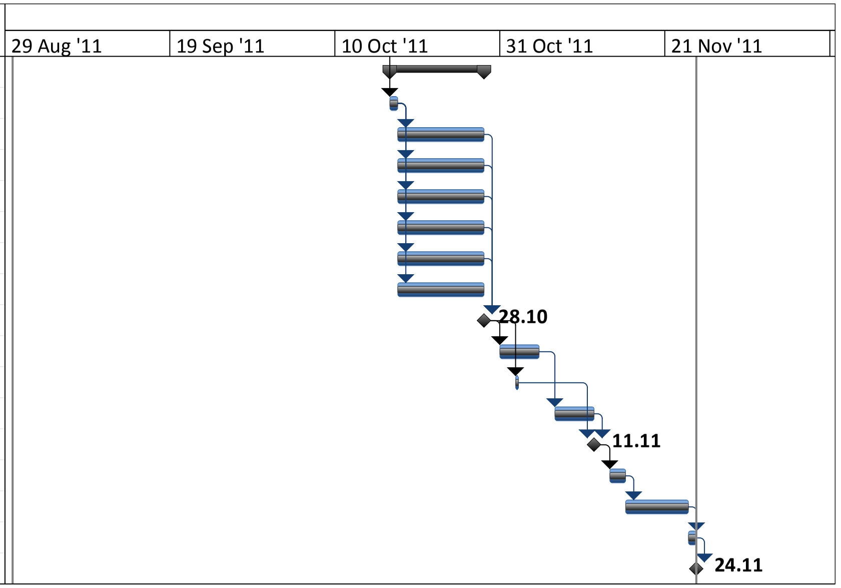


Figure C.2: Gantt chart 2/2

Appendix D

E-mail conversation

D.1 Skype

```
From: Skype Developer Access Support [e-mail address retracted]
Hi,

SkypeKit can be used to develop applications on android devices
(TV's and set top boxes etc) however there is a restriction on
developing for mobile phones and tablets.

There is no waiting list you just need to visit
http://developer.skype.com/signup and then pay the $10 membership fee.

Thanks

[name retracted]
Skype Developer Support
```

D.2 WonderMedia

```
To:  WonderMedia [e-mail address retracted]
Subject:  Specifications for the WonderMedia WM8650
Dear Sir or Madam

I have bought a WonderMedia WM8650 running Android OS 2.2.
However I have not been able to locate the exact specifications for this device,
if you could send them to me it would be much appreciated.

Model number:
WonderMedia WM8650

Build number:
generic-eng 2.2 Froyo v1.2.3-20110409.211203

Sincerely, [name retracted]
```

D.3 Mimo customer support

```
Hello [name retracted] -

Although the Mimo UM-740 has been discontinued for a couple years now, it
uses the same drivers as our current models and so you can download any
drivers you need from our support page of our website.  Note that the
webcam does not require/come with any special webcam driver, and so if
Windows is not recognizing it, possibly it is no longer functioning.
-   -   -
[name retracted]
Mimo Monitors
Suite 15
743 Alexander Road
Princeton, NJ 08540
www.mimomonitors.com

On 9/26/11 7:45 AM, [name retracted] [e-mail address retracted] wrote:

> Hi,
> I’ve lost my driver CD, so I used the drivers listed at the homepage
> (Display and touchscreen drivers) and it seems I cannot get the
> webcamera working, Windows does not recognize it and there is no
> recording device (mic) either.
> Can you steer my in the right direction? The device is Mimo UM-740,
> thanks.
>
> Cheers
```

Appendix E

Ground rules

1. You should be on time for meetings. If you know you're going to be late, let people know. If you don't have a valid excuse for being late, for each started 10 minute session from the start of the meeting, you need to bring a bottle of beer.
2. Unless you're taking notes, it's not allowed to use computers during discussions in group meetings.
3. People should be able to speak freely without being interrupted. If you feel that what you have to say is important, and someone else is talking, write it down and we'll come back to it later.
4. Finish your work before the deadlines. If you can't finish your responsibilities on time and you don't have a valid excuse, you need to buy a beer.
5. The group's *official* language is English. When talking to team members about the project, please speak English.

Appendix F

Project charter

Project:	Interactive Door Sign
Created by:	Michal Krajíček (project manager)
Date:	3. 9. 2011
Phone:	45126315
E-mail:	michalk@stud.ntnu.no
Version no.:	3

Introduction:
Our project, ”Interactive Door Sign”, is solved within the NTNU course TDT4290 Customer Driven Project. Its purpose is to give students the opportunity to go through all the phases of a real software project and gain experience from it. The project is given by IDI, NTNU, and must be solved in close cooperation with their representatives.

Mission:
Our customer – NTNU university, IDI department, has given us a monitor that can be used with a computer running Windows. The mission of this project is to develop a software system that will enable us to use this monitor as an interactive door sign.

Stakeholders:

- Project realization team (Group 7)
- Customer – NTNU IDI represented by Torstein Hjelle (torstein.hjelle@idi.ntnu.no)
- Advisor – Muhammad Asif (muhamma@idi.ntnu.no)

The role of the project team is to fulfill all requirements given by the customer. The team must go through all phases of a typical software project, be able to provide a final report and defend the results in front of the customer and external censors. Our project realization team has 7 members:

Michal Krajíček (michakl@stud.ntnu.no)	Project manager
David Andrássy Eilertsen (davidae@stud.ntnu.no)	Scrum master, Customer contact

Sibel Ayper Taşdemir (tasdemir@stud.ntnu.no)	Document manager, Advisor contact
Knut Esten Melandsø Nekså (knuteste@stud.ntnu.no)	Implementation manager
Andreas Nordahl (andrnord@stud.ntnu.no)	QA manager
Elise Boinnot (emboinno@stud.ntnu.no)	Test manager
Igoris Tirmailovas (igorist@stud.ntnu.no)	Architecture manager

Customer – IDI, NTNU, represented by Torstein Hjelle will provide requirements for the software product.

Advisor – Muhammad Asif will control the work of the project realization team and provide guidance.

Project manager:
Our project manager is Michal Krajíček. He is responsible for governing the whole project. His competences are planning, problem solving, leading group meetings, distribution of work and controlling progress. He is allowed to delegate his powers to other team members.

Objectives:

- Create a working prototype of the application as described by the customer, with all high priority functional requirements implemented.
- The customer has mentioned two main areas to focus on: Security and usability. Different tests will ensure that the prototype meets certain standards regarding both security and usability.
- Create a broad documentation of our work and the prototype.
- Be able to present our final product.

Project scope
The project includes:

- Prototype of the application
- Documentation
- Presentation
- Small database (only for testing and presenting the application)

The project does not include:

- Hardware installation
- Hardware security

Deliverables:

- Software application
- Client software running on the office owner's computer

Budget
The project is non-commercial.

Schedule
The project starts on August 30, 2011 and its duration will be 13 weeks. The final presentation will be on November 24, 2011.

High-Level Work Breakdown Structure:

1. Project directive
2. Project planning
3. Preliminary studies
4. Requirement specifications
5. Architecture studies
6. Scrum sprint #1
7. Scrum sprint #2
8. Project evaluation and conclusion
9. Presentation and demonstration

Communication and meetings:
The team is going to have at least one meeting per week with all members and smaller meetings in compliance with needs. There will also be at least one meeting per week with the advisor. Meetings with the customer will be planned as needed. The communication language will be exclusively English.

Initial project risks:

Risk assessment	Probability	Impact
1. Inaccurate project plan	M	H
2. Lack of motivation	L	H
3. Lack of communication internally	M	H
4. Lack of communication externally	L	H
5. Problems with the implementation	M	M
6. Problems with hardware	M	H
7. Sickness	L	L/M
8. Lack of expertise with SCRUM	L	L

Change control
If the device has limitations so that certain requirements cannot be fulfilled, we should be allowed to change them. All changes to the requirements or the project charter must be approved by the representatives from the implementation team and by the customer.

Sign off

....	....	....
Michal Krajíček	Torstein Hjelle	Muhammad Asif
Project manager	Customer	Advisor

Glossary

BibTeX A tool typically used together with the LaTeX enabling to cite sources in a consistent manner, by separating bibliographic information from the presentation part.

LaTeX Macro package based on TeX – a low-level markup language with high quality typesetting allowing users to focus on content instead of visual part. Its purpose is to simplify TeX typesetting.

Activity diagram An UML notation that describes to workflow of a procedure through a set of activities or actions. These are represented graphically.

Black-box testing Black-box testing is a method of testing software that tests the functionalities of an application. The testers does not need to know, or even understand, the code. They only need to know the possible inputs and the expected outputs associated. The inputs and outputs are based around the requirements.

Burn down chart This is an artifact in SCRUM, showing the remaining work in the sprint backlog, i.e. how much we have done, how much if left and our original assumptions.

Class diagram An UML notation that describes the structure of the system by representing it with it's class, their attributes, most important methods and their relationship to other classes and artifacts in the system.

Communication diagram A notation used in UML to describe the communication between components in the systems, in form of message-passing.

Entity-relationship diagram A tools that is used to describe database modeles, that includes the relationship between two, or more, entities. For example one-to-many and many-to-many relationships.

Gantt chart A chart illustrating a project's schedule. The left part of the chart shows project's WBS and in the right part are start and finish dates of the particular tasks.

Grey-box testing Grey-box testing is a method of testing software. It is a combination of black-box and white-box testing. The testers are not required to have a full access to the code but at least the internal structure.

Model-View-Controller (MVC) MVC is an architectural pattern used in software architecture. This pattern separates the domain logic, the business logic and the user interface permitting independent development, testing and maintenance of each.

MySQL A popular relation database management system, this is used to, for example, create, update, maintain, delete and insert database entries, tables and other structures.

Prototype This a test sample or a concept built into something tangible and testable.

Sprint A term used in SCRUM, an agile software development methodology. A Sprint is equivalent to an iteration in terms of software development management vocabulary.

Subversion (SVN) Subversion is a software versioning and a revision control system distributed under a free license. It is used to maintain current and historical versions of files such as source code, web pages and documentation.

Use-case A description of a requirement can be represented through an use case. It shows the steps one or more actors must perform in order to do something useful with the system.

Waterfall model A sequential development process which flows downwards (you will not have the change to revisit/renew already completed phases of the development).

White-box testing White-box testing is a method of testing software that tests the internal structure of an application. The testers has to understand the code and the system. They also need to have technical knowledge to design and run the tests.

Work breakdown structure A project management method/tool used to break down larger tasks into smaller components/tasks. This makes them more tangible, and easier to controll, measure and schedule.

Acronyms

API Application Programming Interface.

CIVIL Capturing Images and Video In a Library.

CPU Central Processing Unit.

CVS Current Version System.

ERD Entity-relationship diagram.

FPS Frames Per Second.

GUI graphical User Interface.

ICT Information and communications technology.

IDE Integrated Development Environment.

IM Instant Message.

IRC Internet Relay Chat.

JMF Java Media Framework.

RAM Random Access Memory.

RIAs Rich Internet Applications.

RTMFP Real Time Message Flow Protocol.

SDK Software Development Kit.

SIP Session Initiation Protocol.

SUS System Usability Scale.

SVN Subversion.

TSTETIL Time-scaled tasks with explicit task interdependency linkage Gantt chart with linkages among tasks.

WBS Work Breakdown Structure.

WiFi Wireless Fidelity.

XMPP Extensible Messaging and Presence Protocol.